

Program Logic

IBM System/360 Disk Operating System Sort/Merge

Program Number 360N-SM-450

This publication describes the internal logic of the IBM System/360 Disk Operating System Sort/Merge program. It is intended for use by persons involved in program maintenance, and by system programmers who are altering the program design. Program logic information is not necessary for the operation of the program; therefore, distribution of this publication is limited to those with maintenance and alteration requirements.

PREFACE

This Program Logic Manual (PLM) is a detailed guide to the IBM System/360 Disk Operating System Sort/Merge Program; it supplements the program listing by providing descriptive text and flowcharts.

Effective use of this manual requires an understanding of IBM System/360 operation and of IBM System/360 Disk Operating System service programs, assembler language, macro instructions, and sort/merge program specifications. Reference publications for this information are:

- IBM System/360 Principles of Operation, Form A22-6821.
- IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.
- IBM System/360 Disk Operating System, Data Management Concepts, Form C24-3427.
- IBM System/360 Disk Operating System, System Control and System Service Programs, Form C24-3428.
- IBM System/360 Disk Operating System, Supervisor and Input/Output Macros, Form C24-3429.
- IBM System/360 Disk Operating System, System Generation and Maintenance, Form C24-5033.

- IBM System/360 Disk and Tape Operating Systems, Assembler Specifications, Form C24-3414.

Titles and abstracts of other related publications are listed in the IBM System/360 Bibliography, Form A22-6822.

This manual consists of eight major sections. The first section is an introduction to the five major components (phases) of the sort/merge program. The next five sections describe each major component, starting with a general discussion of the phase. Each major component is further subdivided into logical elements (routines). Finally, the function blocks that make up each logical element are described in detail. The seventh section includes the optional routines (data conversion, etc.) that are available for all the phases.

The last section of the manual consists of additional reference material for use in analyzing program details.

The flowcharts for all the major components and for the optional routines are located at the end of the manual. The detailed flowcharts are identified by the letters AA through ZZ. Numerals, such as 00 for the program level flowchart, identify the more general flowcharts.

RESTRICTED DISTRIBUTION: This publication is intended primarily for use by IBM personnel involved in program design and maintenance. It may not be made available to others without the approval of local IBM management.

First Edition, August 1966

Significant changes and additions to the specifications contained in this publication will be reported in subsequent revisions or Technical Newsletters.

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

A form is provided at the back of this publication for readers' comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Endicott, New York 13760.

CONTENTS

INTRODUCTION	9	Output Routine - BG.	77
Program Organization	12	End-of-Phase Routine - BH.	80
Program Generation	12	Initialization for ADDROUT Run - BJ.	81
Program Characteristics.	13	Input Routine for ADDROUT Run - BK, BL	82
System Requirements.	22	Internal Sort (ADDROUT Run) - BF	84
ASSIGNMENT PHASE (PHASE 0) - 01.	23	Output Routine (ADDROUT Run) - BG.	84
Read and Compress Control Cards - AA	25	Multi-Volume, Exit 11 Linkage - BM, BN	84
Sort Compressed Control Cards - AB	30	PHASE 2 - EXTERNAL SORT OR MERGE	86
Scan SORT, MERGE, and RECORD Control Cards - AC.	31	Phase 2 Initialization, Fixed-Length Records - CA.	96
Scan INPFIL, OUTFIL, MODS, OPTION and END Control Cards - AD, AE.	33	Pass-Pass Routine, Fixed-Length Records - CC.	100
Open Work Area and Pre-Edit - AF, AG.	35	Input Routine, Fixed-Length Records - CD.	101
Open Work Area and Pre-Edit (continued) - AG.	38	Sequence G Compare Loop, Fixed-Length Records - CE.	103
Compute Maximum Allowable Input and Output Record and Block Lengths -- AH	42	Sequence F Compare Loop, Fixed-Length Records - CF.	104
Post Edit - AJ	45	Sequence E Compare Loop, Fixed-Length Records - CG.	105
Compute Constants for Fixed-Length Records - AK, AL.	50	Sequence D Compare Loop, Fixed-Length Records - CH.	106
Compute Constants (Variable-Length Records) - AM	55	Sequence C Compare Loop, Fixed-Length Records - CJ.	107
Compute Constants for ADDROUT Sort - AN.	58	Sequence B Compare Loop, Fixed-Length Records - CK.	107
Select Order of Merge - AP, AQ	61	Output Routine, Fixed-Length Records - CL.	108
Print Control Card and Fetch Next Phase - AR, AS.	64	Calculate Interleaved Disk Address Routine, Fixed-Length Records - CM.	110
INTERNAL SORT (PHASE 1) - 02	68	Phase 2 Initialization, Variable-Length Records - CN.	110
Initialization Routine for Multi-Volume (Exit 11 Linkage) - BA	71	Merge-Merge Routine, Variable-Length Records - CP.	114
Format Routine - BB.	72	Pass-Pass Routine, Variable-Length Records - CQ.	117
Initialization for Disk or Tape Input - BC.	72	Input Routine, Variable-Length Records - CR.	119
Input Routine for Disk or Tape - BD.	74		
Tape Input Routine - BE.	75		
Internal Sort - BF	75		

Sequence F Compare Loop, Variable-Length Records - CS.120	Error Routine - ED178
Sequence E Compare Loop, Variable-Length Records - CT.121	File D Compare Loop - EE180
Sequence D Compare Loop, Variable-Length Records - CU.122	File C Compare Loop - EF181
Sequence C Compare Loop, Variable-Length Records - CV.123	File B Compare Loop - EG182
Sequence B Compare Loop, Variable-Length Records - CW.125	Output Routine - EH.182
Output Routine, Variable-Length Records - CX.126	End-of-Job Routine - EJ.187
Calculate Interleaved Disk Address Routine, Variable-Length Records - CY .128		End-of-Job Messages - EK187
FINAL MERGE (PHASE 3) - 04131	Sequence-Error Routine - EL.188
Phase 3 Initialization, Fixed-Length Records - DA.133	Open/Close Routine - EM.188
Input Routine, Fixed-Length Records - DB.136	Checkpoint Routine - EP.192
Mainline Compare Routine, Fixed-Length Records - DC,DD,DE,DF,DG.138	OPTIONAL ROUTINES.194
Compute Input Interleaved Disk Address, Fixed-Length Records - DH. . .141		Relocator Routine - FA194
Output Routine, Fixed-Length Records - DJ,DK143	Fixed-Point Convert/Reconvert Routine - FB.196
Label-Linkage Routine (LLR), Fixed-Length Records - DL147	Floating-Point Convert/Reconvert Routine - FC.196
Phase 3 Initialization, Variable-Length Records - DM.148	Packed-Decimal (SIGPAK) Convert/Reconvert Routine - FD.197
Input Routine, Variable-Length Records - DN.151	Zoned-Decimal (SIGZON) Convert/Reconvert Routine - FE.197
Mainline Compare Routine, Variable-Length Records - DP, DQ, DR, DS.153	Equal Routine - FF198
Compute Input Interleaved Disk Address, Variable-Length Records - DT .157		APPENDIX A: CONSTANTS AND ABBREVIATIONS301
Output Routine, Variable-Length Records - DU, DV.159	Constants.301
Label-Linkage Routine (LLR), Variable-Length Records - DW.164	Definitions of Abbreviations349
MERGE ONLY (PHASE 4) - 05.167	APPENDIX B: LABEL REFERENCES350
Initialize Open/Close Routine - EA . . .169		Assignment Phase350
Initialize Mainline - EB171	Phase 1.354
Input Routine - EC176	Phase 2.356
		Phase 3.358
		Phase 4.360
		Optional Routines.363
		APPENDIX C: EXPLANATION OF FLOWCHART SYMBOLS364
		APPENDIX D: ERROR MESSAGES365
		GLOSSARY373
		INDEX.374

Chart 00. Program Level Design.	8	Chart 03. External Sort or Merge, Fixed-Length Records (Phase 2), DSORT201 or DSORT202.232
Chart 01. Assignment Phase (Phase 0).200	Chart CA. Phase 2 Initialization, Fixed-Length Records.233
Chart AA. Read and Compress Control Cards, DSORT.201	Chart CB. Merge-Merge Routine, Fixed-Length Records.234
Chart AB. Sort Compressed Control Cards, DSORT002202	Chart CC. Pass-Pass Routine, Fixed-Length Records.235
Chart AC. Scan SORT, MERGE, and RECORD Control Cards, DSORT003.203	Chart CD. Input Routine, Fixed-Length Records236
Chart AD. Scan INPFIL, OUTFIL, MODS, OPTION and END Control Cards, DSORT004.204	Chart CE. Sequence G Compare Loop, Fixed-Length Records.237
Chart AE. Scan INPFIL, OUTFIL, MODS, OPTION and END Control Cards (Cont'd), DSORT004.205	Chart CF. Sequence F Compare Loop, Fixed-Length Records.238
Chart AF. Open Work Area and Pre-edit, DSORT005.206	Chart CG. Sequence E Compare Loop, Fixed-Length Records.239
Chart AG. Open Work Area and Pre-edit (Cont'd), DSORT005.207	Chart CH. Sequence D Compare Loop, Fixed-Length Records.240
Chart AH. Compute Maximum Allowable Input and Output Record and Block Lengths, DSORT006208	Chart CJ. Sequence C Compare Loop, Fixed-Length Records.241
Chart AJ. Post Edit, DSORT007209	Chart CK. Sequence B Compare Loop, Fixed-Length Records.242
Chart AK. Compute Constants, DSORT008210	Chart CL. Output Routine, Fixed-Length Records.243
Chart AL. Compute Constants (Fixed-Length Records), DSORT008.211	Chart CM. Calculate Interleave Disk Address, Fixed-Length Records244
Chart AM. Compute Constants (Variable-Length Records), DSORT008212	Chart 03. External Sort or Merge, Variable-Length Records (Phase 2), DSORT203 or DSORT204.245
Chart AN. Compute Constants (ADDROUT), DSORT008213	Chart CN. Phase 2 Initialization, Variable-Length Records246
Chart AP. Select Order of Merge, DSORT009.214	Chart CP. Merge-Merge Routine, Variable-Length Records247
Chart AQ. Select Order of Merge (Cont'd), DSORT009.215	Chart CQ. Pass-Pass Routine, Variable-Length Records248
Chart AR. Print Option and Fetch Next Phase, DSORT010216	Chart CR. Input Routine, Variable-Length Records249
Chart AS. Print Option and Fetch Next Phase (Cont'd), DSORT010.217	Chart CS. Sequence F Compare Loop, Variable-Length Records250
Chart 02. Internal Sort (Phase 1)218	Chart CT. Sequence E Compare Loop, Variable-Length Records251
Chart BA. Initialization for Multi-Volume (Exit 11 Linkage), DSORT101.219	Chart CU. Sequence D Compare Loop, Variable-Length Records252
Chart BB. Format Routine, DSORT102.220	Chart CV. Sequence C Compare Loop, Variable-Length Records253
Chart BC. Initialization Routine for Disk or Tape Input, DSORT103.221	Chart CW. Sequence B Compare Loop, Variable-Length Records254
Chart BD. Input Routine for Disk or Tape Input, DSORT103.222	Chart CX. Output Routine, Variable-Length Records255
Chart BE. Tape Input Routine, DSORT103.223	Chart CY. Calculate Interleave Disk Address, Variable-Length Records.256
Chart BF. Internal Sort, DSORT103224	Chart 04. Final Merge, Fixed-Length Records (Phase 3), DSORT301 or DSORT302.257
Chart BG. Output Routine, DSORT103.225	Chart DA. Phase 3 Initialization, Fixed-Length Records.258
Chart BH. End-of-Phase Routine, DSORT105.226	Chart DB. Input Routine, Fixed-Length Records259
Chart BJ. Initialization for ADDRROUT Run, DSORT104227	Chart DC. Mainline Compare Routine, Fixed-Length Records.260
Chart BK. ADDRROUT Run Input Routine, DSORT104.228		
Chart BL. ADDRROUT Run Input Routine (Cont'd), DSORT104.229		
Chart BM. Multi-Volume Routine, DSORT103.230		
Chart BN. Multi-Volume Routine (Cont'd), DSORT103.231		

Chart DD. Mainline Compare Routine Fixed-Length Records (Cont'd)	Chart 05. Merge-Only (Phase 4), DSORT401 and DSORT402
.261	.280
Chart DE. Mainline Compare Routine Fixed-Length Records (Cont'd)	Chart EA. Initialize Open/Close Routine, DSORT401
.262	.281
Chart DF. Mainline Compare Routine Fixed-Length Records (Cont'd)	Chart EB. Initialize Mainline, DSORT402.
.263	.282
Chart DG. Mainline Compare Routine Fixed-Length Records (Cont'd)	Chart EC. Input Routine, DSORT402
.264	.283
Chart DH. Compute Input Interleaved Disk Address, Fixed-Length Records.	Chart ED. Error Routine, DSORT402
.265	.284
Chart DJ. Output Routine, Fixed-Length Records.	Chart EE. File D Compare Loop, DSORT402.
.266	.285
Chart DK. Output Routine, Fixed-Length Records (Cont'd)	Chart EF. File C Compare Loop, DSORT402.
.267	.286
Chart DL. Label Linkage Routine (LLR), Fixed-Length Records	Chart EG. File B Compare Loop, DSORT402.
.268	.287
Chart 04. Final Merge, Variable-Length Records (Phase 3), DSORT303 or DSORT304.	Chart EH. Output Routine, DSORT402.
.269	.288
Chart DM. Phase 3 Initialization, Variable-Length Records	Chart EJ. End of Job Routine, DSORT402.
.270	.289
Chart DN. Input Routine, Variable-Length Records	Chart EK. End of Job Messages, DSORT401.
.271	.290
Chart DP. Mainline Compare Routine, Variable-Length Records	Chart EL. Sequence Error Routine, DSORT401.
.272	.291
Chart DQ. Mainline Compare Routine, Variable-Length Records (Cont'd).	Chart EM. Open/Close Routine, DSORT401.
.273	.292
Chart DR. Mainline Compare Routine, Variable-Length Records (Cont'd)	Chart EN. Open/Close Routine (Cont'd), DSORT401.
.274	.293
Chart DS. Mainline Compare Routine, Variable-Length Records (Cont'd).	Chart EP. Checkpoint Routine, DSORT402.
.275	.294
Chart DT. Compute Input Interleaved Disk Address, Variable-Length Records .276	Chart FA. Relocator Routine, Phases 2, 3, and 4
.276	.295
Chart DU. Output Routine, Variable-Length Records	Chart FB. Fixed Point Convert/Reconvert
.277	.296
Chart DV. Output Routine, Variable-Length Records (Cont'd).	Chart FC. Floating-Point Convert/Reconvert
.278	.297
Chart DW. Label Linkage Routine (LLR), Variable-Length Records.	Chart FD. Packed-Decimal (SIGPAK) Convert/Reconvert
.279	.298
	Chart FE. Zoned-Decimal (SIGZON) Convert/Reconvert
	.299
	Chart FF. Equal Routine, Phases 2, 3, and 4
	.300

Figure 1. Program I/O Flow for Sort	10	Figure 30. Main Storage Output Area	90
Figure 2. Program I/O Flow for Merge-Only.	11	Figure 31. Compare Tree	91
Figure 3. Interleaved Method of Output.	14	Figure 32. Interleaving (3-Way Merge)	92
Figure 4. Summary of Sort/Merge Control Statements.	15	Figure 33. Phase 2 Main Storage Layout for Fixed-Length Records	94
Figure 5. Disk Input/Output Blocking Formats	16	Figure 34. Phase 2 Main Storage for Variable-Length Records	95
Figure 6. Tape Input/Output Blocking Formats	17	Figure 35. Calculate Interleaved Disk Address	111
Figure 7. Control Field Formats	18	Figure 36. Variable-Length Disk Address - Block Count	116
Figure 8. Tape Input/Output Scheme for Sort or Merge-Only Operation.	22	Figure 37. Calculate Interleaved Disk Address	130
Figure 9. Assignment Phase Main-Storage Layout	24	Figure 38. Phase 3 Main Storage Layout.	132
Figure 10. Sort Compressed Control Cards	30	Figure 39. Rewind Action Taken at End-of-Volume Time for Multi-Volume Tape Files.	134
Figure 11. Work Area Table (2 Sections)	35	Figure 40. Contents of Registers at Fetch Time.	136
Figure 12. Work Area Table (5 Sections)	36	Figure 41. Compare Tree (for a 4-Way Merge).	141
Figure 13. Disk Sort/Merge Phase Size Formulas.	41	Figure 42. Calculate Interleaved Disk Address	142
Figure 14. Exit Chart	50	Figure 43. Rewind Action Taken at End-of-Volume Time for Multi-Volume Tape Files.	150
Figure 15. Sort Area Layout - Overlay 8	54	Figure 44. Contents of Registers at Fetch Time.	152
Figure 16. Phase 1 Internal Sorting (Ascending Sequence).	69	Figure 45. Compare Tree (for a 4-Way Merge).	155
Figure 17. Phase 1 Main Storage Layout.	70	Figure 46. Calculate Interleaved Disk Address	158
Figure 18. Disk Address Format.	71	Figure 47. Maximum Use of Output Track	161
Figure 19. Limits of Input Area	73	Figure 48. Phase 4 Main Storage Layout.	168
Figure 20. Doublets	75	Figure 49. Maximum Use of Output Track	184
Figure 21. 4-Record Sequences	76	Figure 50. Linkages Between Overlays.	193
Figure 22. 8-Record Sequences	76	Figure 51. Floating-Point Conversion.	197
Figure 23. Output	76	Figure 52. Phase 2 Disk Address Table	324
Figure 24. Address Creation	78	Figure 53. Phase 2 Limits Table	327
Figure 25. Output, Variable-Length Records	79	Figure 54. Phase 3 Disk Input Address Table	333
Figure 26. Phase 1 Compression.	80		
Figure 27. Sort Blocks.	87		
Figure 28. Phase 2 Merge.	88		
Figure 29. Variable-Length Record Input Area.	89		

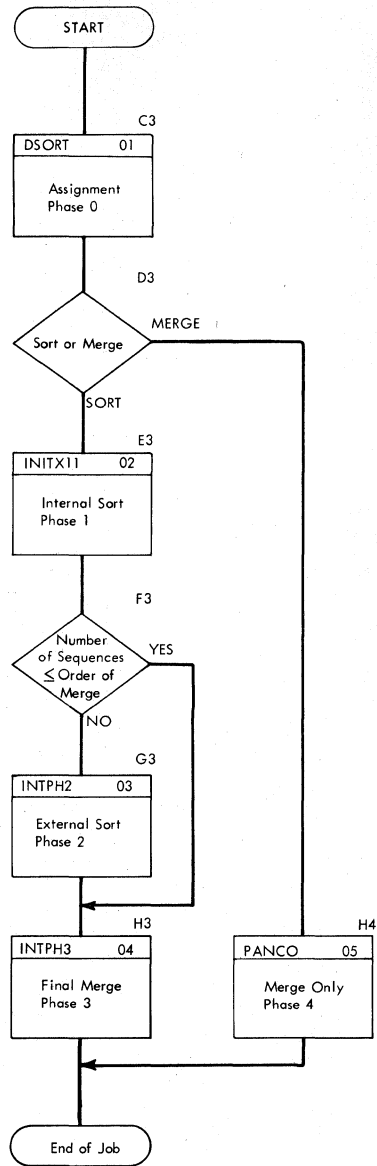


Chart 00. Program Level Design

The IBM System/360 Disk Operating System Sort/Merge Program enables the user to sort multiple files of random records, or merge multiple files of sequenced records, into one sequential file. This program is designed to meet the sorting and merging requirements of disk-oriented installations with 16K to 512K bytes of main storage. It is a generalized program that must be incorporated into the resident version of the Disk Operating System library. The program is designed for operation in a disk-resident operating system environment under the supervision of a control program.

At execution time, control statements (see Program Characteristics) will tailor the generalized sort/merge program to the user's specific application. The sort/merge control statements are punched into cards and inserted into the card reader with the symbolic address SYSIPT. All required user-prepared job-control statements are read by the device (either a card reader or a magnetic tape unit) assigned to SYSRDR. The sort/merge program will be retrieved in overlays from the disk-resident core image library by the control program.

Control-data information may be contained in as many as 12 fields in each record. The program assumes that input records for a sort operation are in random sequence. Records can be sorted or merged into ascending or descending sequence, and an individual sequence can be specified for each control-data field. The output sequence for a merge-only operation must be the same as the input sequence.

The sort/merge program is divided into five phases (see Chart 00):

Assignment Phase (Phase 0)
 Internal-Sort Phase (Phase 1)
 External-Sort Phase (Phase 2)
 Final Merge Phase (Phase 3)
 Merge-Only Phase (Phase 4)

If sorting is to be done, the assignment phase and phases 1, 2, and 3 are executed. If only merging is to be done, the assignment and merge-only phases are executed. (See Figures 1 and 2 for I/O flow diagrams.)

The sort/merge program:

- Translates mnemonic sort/merge

control-card information that describes the file parameters for each input and output file.

- Sorts multiple input files.
- Allows multivolume input and/or output for tape and disk.
- Provides for input from and output to disk storage (IBM 2311 Disk Storage Drive(s) only), or 7- or 9-track magnetic tapes (see System Requirements for possible combinations of 7- and 9-track tapes). Disk and tape input and output can be distributed over multiple drives.
- Merges up to four tape and/or disk input files.
- Provides for writing on disk or tape an output file that consists of the disk addresses of the sorted records (ADDRROUT=A option) or disk addresses plus control data of the sorted records (ADDRROUT=D option).
- Provides for determining the number of work area tracks required for a sort operation (CALCAREA option).
- Provides for specification of alternate input tape drives and alternate output tape drives for either a sort or a merge operation.
- Provides checkpoint, interrupt, and restart procedures for sort operation.
- Provides exits and storage areas for user-written routines.
- Prints out control-card information (optional), record counts at the end of phases 1, 3, and 4, and necessary diagnostics.
- Provides, for sorting, the option to bypass unreadable data blocks when the input file is being read from tape, or indicates the need for operator intervention when the input file is being read from tape or disk.
- Provides, for the merge-only operation, the option to bypass unreadable data blocks when the input files are being read from tape or disk, or to indicate the need for operator intervention.

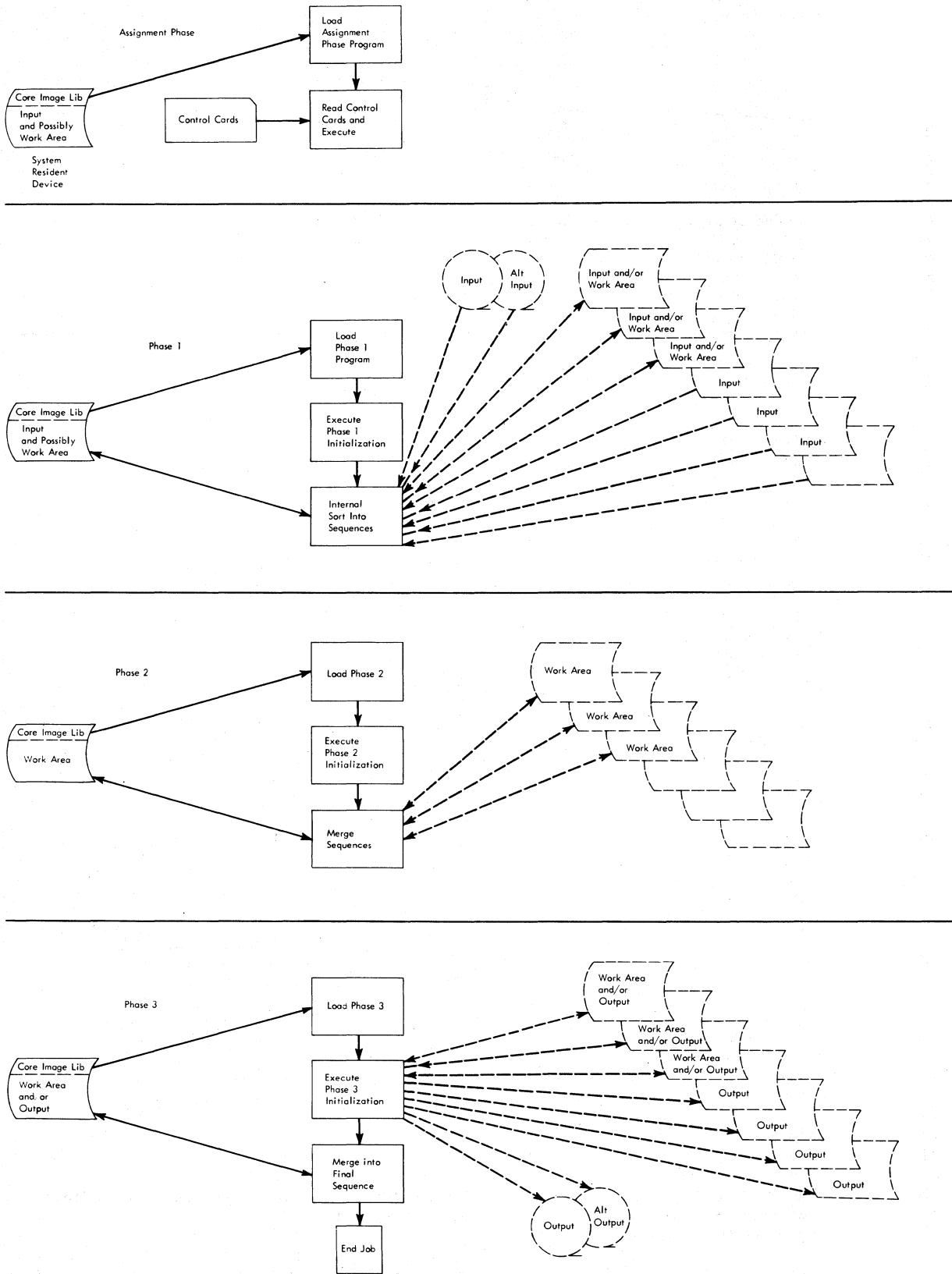


Figure 1. Program I/O Flow for Sort

Assignment Phase

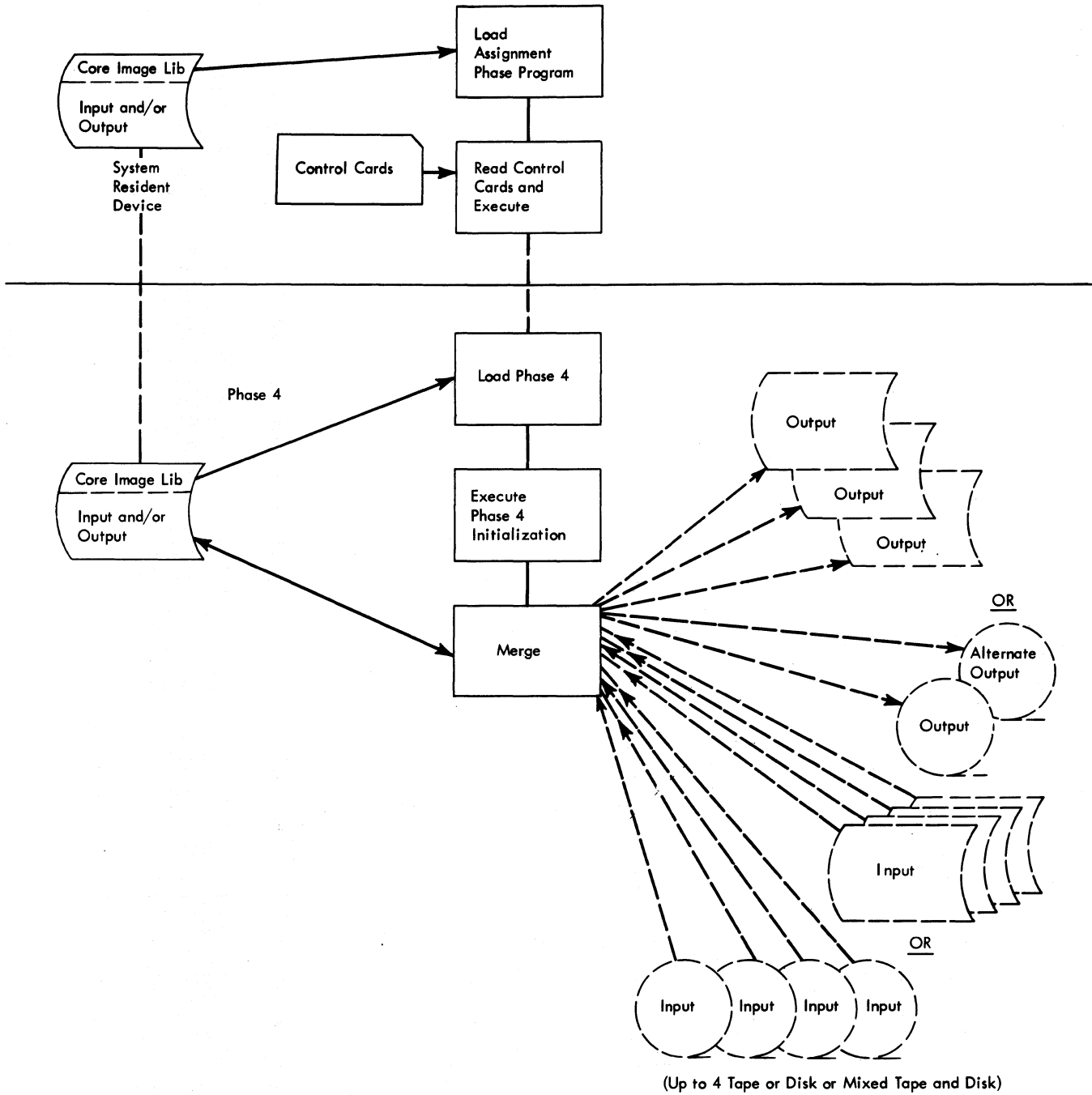


Figure 2. Program I/O Flow for Merge-Only

- Checks the sequence of the records during the final output pass of a sort (phase 3) or during a merge (phase 4).
- Permits the user to specify either ascending or descending sequence for each individual control-data field.
- Processes standard System/360 volume and file labels.
- Provides exits for user processing of non-standard labels (on tape) and user labels (on tape or disk).

- Permits mixed labels for tape input to a merge-only operation.

PROGRAM ORGANIZATION

ASSIGNMENT PHASE (PHASE 0)

The assignment phase:

1. Reads and stores data from the sort/merge control statements.
2. Performs a diagnostic check for missing sort/merge control statements and duplicate or invalid field definers.
3. Prints out the sort/merge control statements and parameters if the PRINT option is specified.
4. Converts the parameters to binary format.
5. Calculates and stores the constants required by the following phases.

INTERNAL-SORT PHASE (PHASE 1)

Phase 1 performs the initial sequencing of the input file(s). The records are read into the main-storage input area and sorted into sequences that are at least as long as the number of records that can be sorted internally at one time. These sequences are written in either the first or the second half of the disk-storage work area. The output is interleaved to minimize the time required to complete the sort.

Phase 1 sorts multiple input files (these files may also be multivolume). Exits from the program are provided to allow processing and/or label-checking by user-prepared routines. The checkpoint, interrupt, and restart procedure is provided at the end of the phase.

EXTERNAL SORT PHASES (PHASE 2 AND PHASE 3)

The external sort phases merge the ordered sequences produced by the internal sort phase. They perform a 2-way to 7-way merge on the record sequences produced by the internal sort phase.

Phase 2 repeatedly merges the sequences from the 2311 disk-storage work area until their number is equal to or less than the

program-determined order of merge. Phase 3 performs the last pass of the external sort. If the number of sequences produced by phase 1 is equal to or less than the order of merge, phase 2 is bypassed and only phase 3 is executed.

Both phases provide the checkpoint, interrupt, and restart feature. Phase 2 uses the interleaved output technique; phase 3 allows multivolume output files and provides exits from the program to allow user modifications by user-prepared routines.

MERGE-ONLY PHASE (PHASE 4)

The merge-only phase is used to merge existing presorted files into one sequential file. A maximum of four input files can be merged and these may be on disk and/or tape in any combination. The program allows for multiple-volume input and output files. The program also allows a single file to be reblocked and sequence-checked. Exits from the program are provided to allow for user modifications by user-prepared routines.

PROGRAM GENERATION

The sort/merge program resides in the relocatable library and consists of three entities:

- Primary processor generation modules
- Intermediate processor generation modules
- Relocatable object modules

These modules are designed to enable the user to linkage-edit into the core image library only those modules required to formulate a generalized sort/merge program tailored to the user's specific job applications. At system generation time, the user has the choice of linkage editing any one of seven distinct sort/merge object programs into the core image library. After the specific object program has become resident in the core image library, the user can delete all the modules from the relocatable library.

The programs that can be generated at linkage-edit time are:

1. The entire sort/merge program.

2. A program that will (1) sort fixed-length records, and (2) perform the ADDRROUT option for fixed- or variable-length records.
3. A program that will (1) sort variable-length records, and (2) perform the ADDRROUT option for fixed- or variable-length records.
4. A program that will (1) sort fixed- or variable-length records, and (2) perform the ADDRROUT option for fixed- or variable-length records.
5. A program that will merge fixed- or variable-length records.
6. A program that will (1) sort fixed-length records, (2) merge fixed- or variable-length records, and (3) perform the ADDRROUT option for fixed- or variable-length records.
7. A program that will (1) sort variable-length records, (2) merge fixed- or variable-length records, and (3) perform the ADDRROUT option for fixed- or variable-length records.

For further details about program generation for sort/merge, see IBM System/360 Disk Operating System System Generation and Maintenance, Form C24-5033.

PROGRAM CHARACTERISTICS

INTERLEAVED OUTPUT

Interleaving is a technique for writing sequences in the disk work area in a manner that will minimize seek time during each subsequent pass. It may be used in a system with only one disk device but is most efficient when used with two or more disk devices. The disk work area is formatted for interleaved output during phase 1 and records are blocked for output before being written in the work area, except when there is only one sort block per track.

The interleave factor (sometimes called the gap factor) is equal to the order of

merge at the start of a pass and may be reduced in the late stages of the pass. Figure 3 is an example of interleaved output when the order of merge is four. The blocks that constitute the first sequence are written at every fourth address, starting with the first address. The blocks that constitute the second sequence are then written at every fourth address, starting with the second address. After the first four sequences have been written, the allotted disk work area (in this example, $4 \times 3 = 12$ blocks) will have been filled and the pattern is then repeated every four sequences. (For a more detailed description of interleaving, see the introduction to phase 2.)

JOB CONTROL STATEMENTS

Standard job-control cards are required to define a sort or merge operation to the job-control program. These cards are:

- General--JOB, ASSGN, DATE, EXEC
- Input File(s)--VOL, DLAB or TPLAB, XTENT
- Work Area--VOL, DLAB, XTENT
- Output File--VOL, DLAB or TPLAB, XTENT

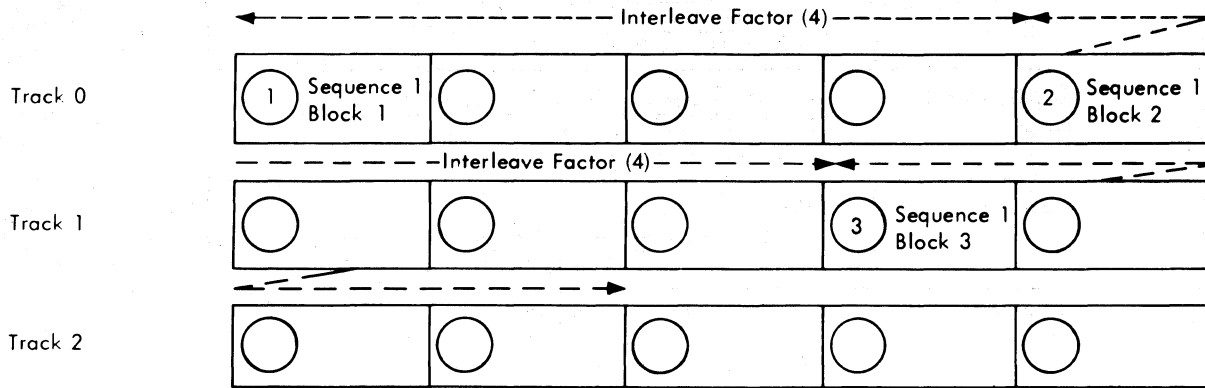
For a complete discussion of the job-control cards and their formats, see IBM System/360 Disk Operating System, System Control and System Service Programs, Form C24-3428, and IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.

Note: For merge-only operations, the program requires two consecutive disk tracks to temporarily store overlay DSORT401 and a portion of DSORT402 as they exchange residence in main storage at various points in the program. The user must define this area as at least a 2-track work area.

The job-control statements are followed by the sort/merge control statements.

Example: Number of Blocks in an Output Sequence = 3
 Order of Merge (M) = 4
 Blocks per Tracks Formatted by Phase 1 = 5

Note: Numbers in circles indicate the order in which blocks are written.



After All Sequences Have Been Written, Work Area is Shown Below

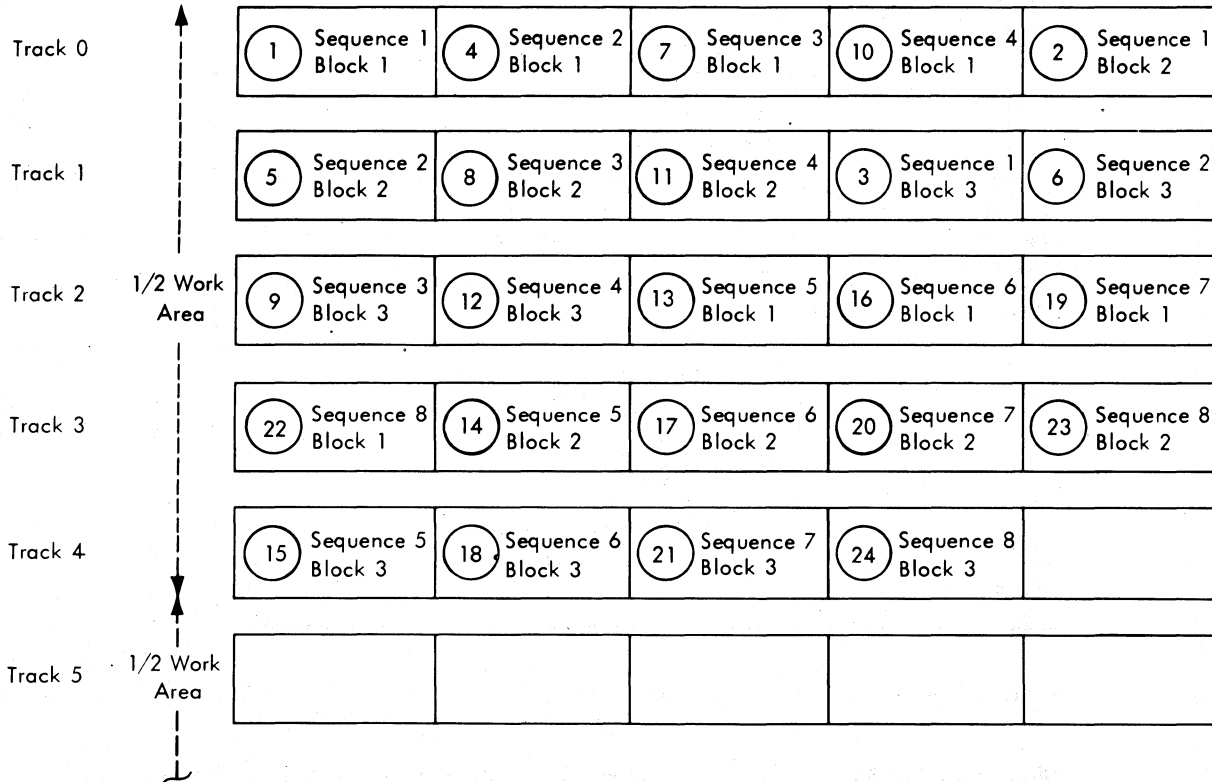


Figure 3. Interleaved Method of Output

SORT/MERGE CONTROL STATEMENTS

Control statements are necessary to define the user's specific sort or merge operation. The user must describe the

files to be sorted or merged, the control-data fields, the program options to be used, and the modifications to be made to the program. (See Figure 4, Summary of Sort/Merge Control Statements.) This information is punched into

STATEMENT DEFINER	STATUS	OPERAND DEFINER, STATUS, AND VALUE FORMAT
SORT	Required for sort runs	FIELDS (Required) = (P1, M1, S1 ... P12, M12, S12) FORMAT (Required) = xx FILES (Optional) = m SIZE (Required) = n
MERGE	Required for merge runs	FIELDS (Required) = (P1, M1, S1 ... P12, M12, S12) FORMAT (Required) = xx FILES (Required) = m
RECORD	Required	TYPE (Required) = x LENGTH (Required) = (L1, L2, L3) for FLR = (L1, L2, L3, L4, L5) for VLR
INPFIL	Required	INPUT (Required) = X for sort = (Xa, Xb, Xc, Xd) for merge-only VOLUME (Optional) = (Na, Nb, Nc, ... Ni) BLKSIZE (Required) = (n, X) OPEN (Optional) = n CLOSE (Optional) = n BYPASS (Optional)
OUTFIL	Required	BLKSIZE (Required) = n OUTPUT (Required) = x OPEN (Optional) = n CLOSE (Optional) = n NOTPMK (Optional)
MODS	Required if user-written subroutines will be added	PH1 = (Name, Address, E11, E12, E13) PH3 = (Name, Address, E31, E32) PH4 = (Name, Address, E41, E42, E43, E44, E45)
OPTION	Optional	PRINT (Optional) STORAGE (Optional) = n VERIFY (Optional) KEYLEN (Optional) = x RESTART (Optional) ADDROUT (Optional) = x CALCAREA (Optional) LABEL (Optional) = (O, I) (O, Ia, Ib, Ic, Id)
END	Required	

Figure 4. Summary of Sort/Merge Control Statements

control-statement cards, and these cards are inserted into the input stream at SYSIPT.

During the assignment phase, each control statement is checked for incorrect entries and inconsistent combinations of entries. If any errors are detected, the program prints a message that indicates the nature of each error. When all the control statements have been checked, if no abort-type errors were detected the user is

given the option to correct the errors and restart or to cancel the job.

If certain optional entries are not made, the assignment phase will assume predetermined parameters.

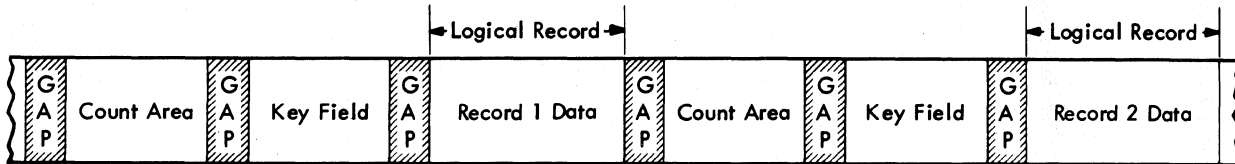
For a complete discussion of control statements and control card format, see IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.

RECORD FORMAT

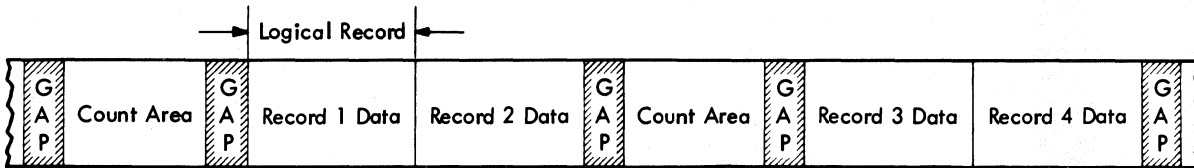
The sort/merge program can process fixed-length or variable-length records. These records may be unblocked or blocked in fixed- or variable-length blocks (see Figures 5 and 6). The most efficient input file record format is fixed-length records in fixed-length blocks, and the least efficient is variable-length records unblocked.

The KEY field, on disk input files, is used by the sort/merge program only with fixed-length unblocked records. The KEY field in all other formats is bypassed.

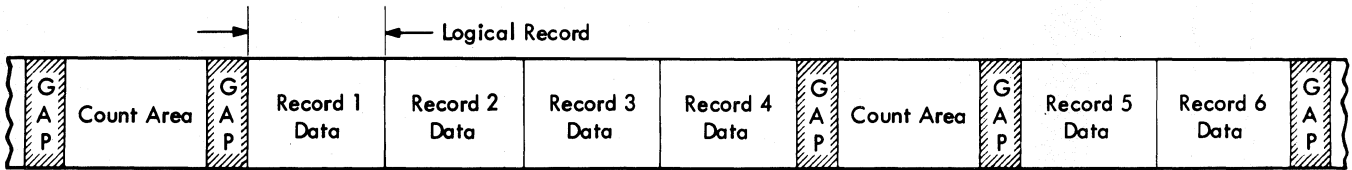
For a merge-only operation, the sort/merge program will accept input from mixed storage media (tape and disk). However, any one file must reside completely on disk or tape. There is no restriction on how the input media can be mixed.



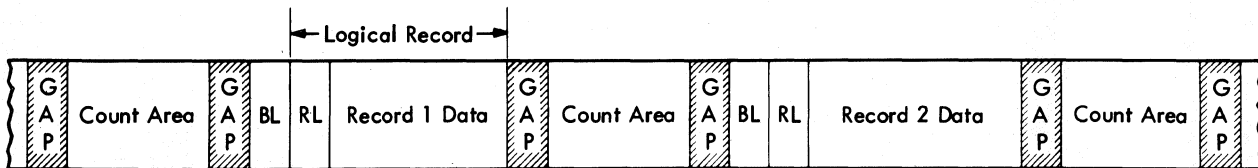
Unblocked Fixed - Length Records (Input/Output) Key Field may be omitted



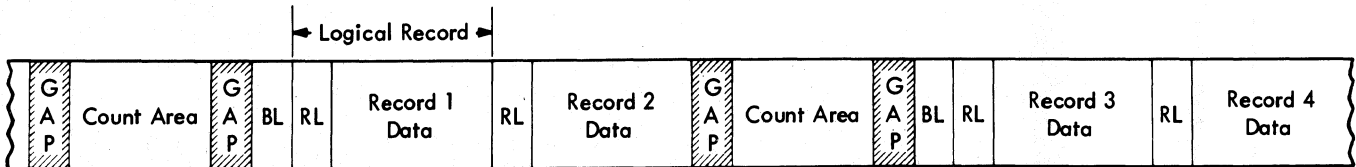
Fixed - Length Records in Fixed - Length Blocks (Input/Output)



Fixed - Length Records in Variable - Length Blocks (Input)



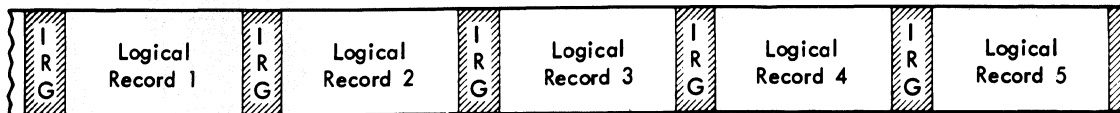
Variable Length -- Unblocked Records (Input/Output)



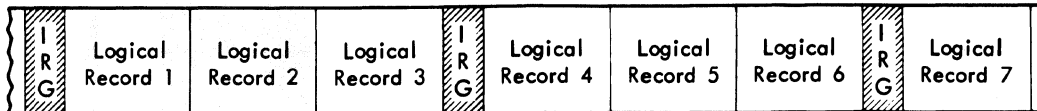
Variable Length -- Blocked Records (Input/Output)

BL (Block Lengths) = The number of bytes between gaps separating count areas.

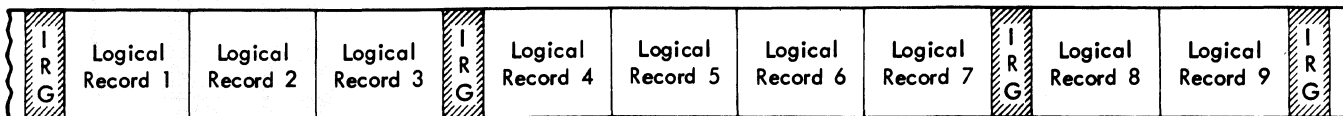
Figure 5. Disk Input/Output Blocking Formats



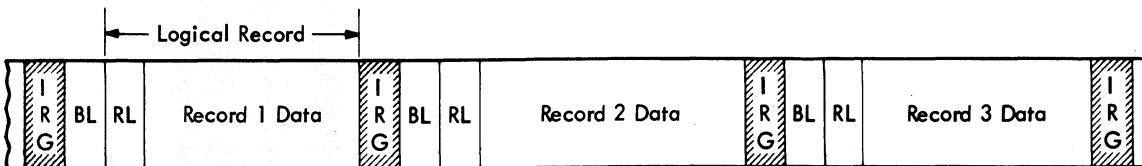
Unblocked Fixed - Length Records (Input/Output)



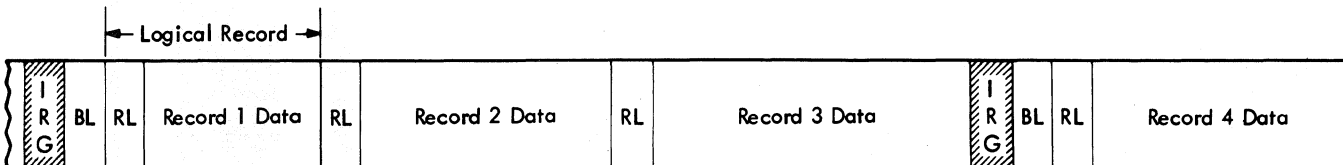
Fixed - Length Records in Fixed - Length Blocks (Input/Output)



Fixed - Length Records in Variable - Length Blocks (Input)



Unblocked Variable - Length Records (Input/Output)



Blocked Variable - Length Records (Input/Output)

BL (Block Length) = The number of bytes between interrecord gaps.

Figure 6. Tape Input/Output Blocking Formats

For a complete discussion of record formats, see IBM System/360 Disk Operating System, Data Management Concepts, Form C24-3427 and IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.

CONTROL-DATA FIELDS

A control-data field is a group of adjacent bytes within a data record. The program is capable of sorting or merging records with up to 12 control-data fields with a maximum total length of 256 bytes.

The most significant field is the major control field, and is always compared first. All other fields are minor fields, and are compared according to their relative significance. A given minor control field is compared only if the major control fields are compared and found to be equal.

The individual fields may be adjacent or separated; if they contain only unsigned binary data, they may overlap (see Figure 7). They may occur anywhere within a data record except in the record-length field at the beginning of each variable-length record.

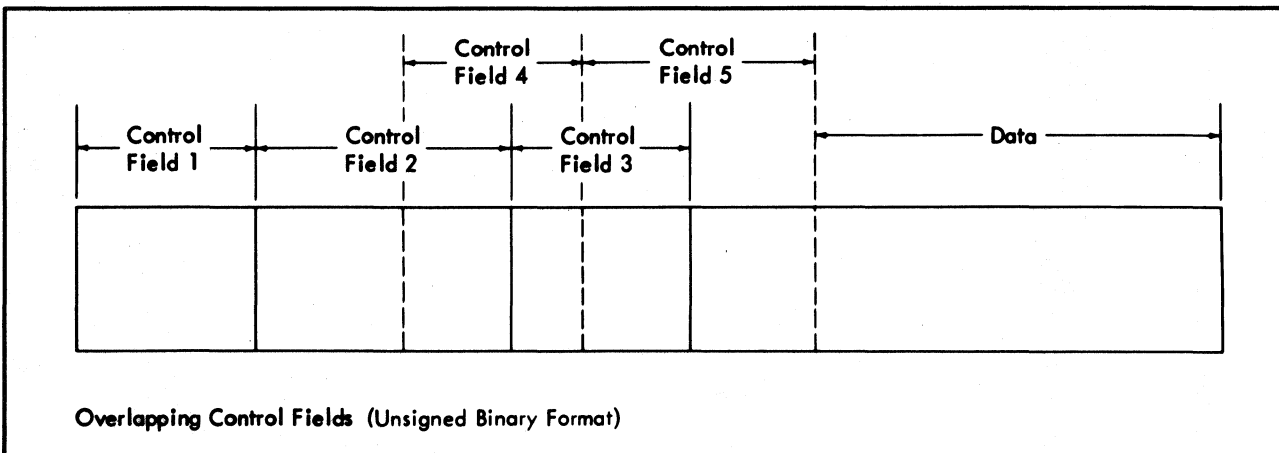
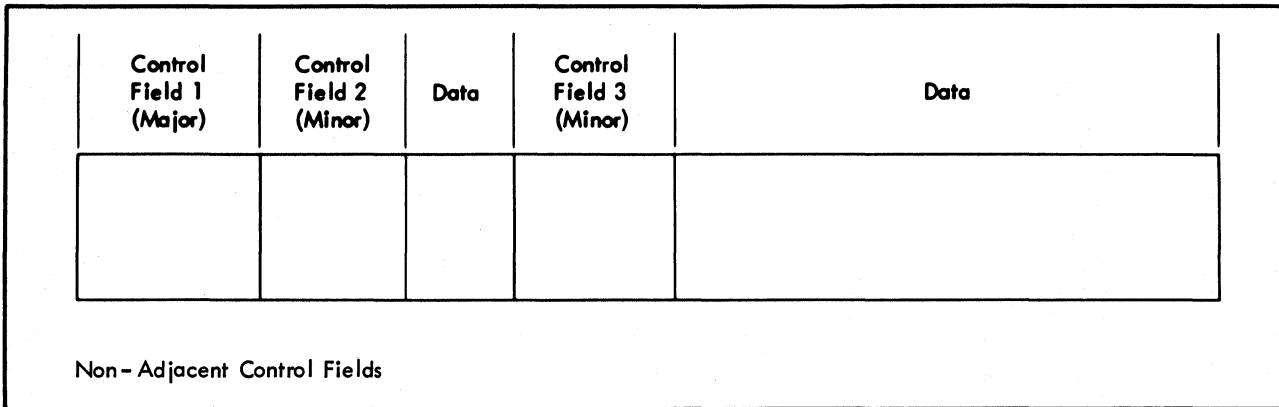
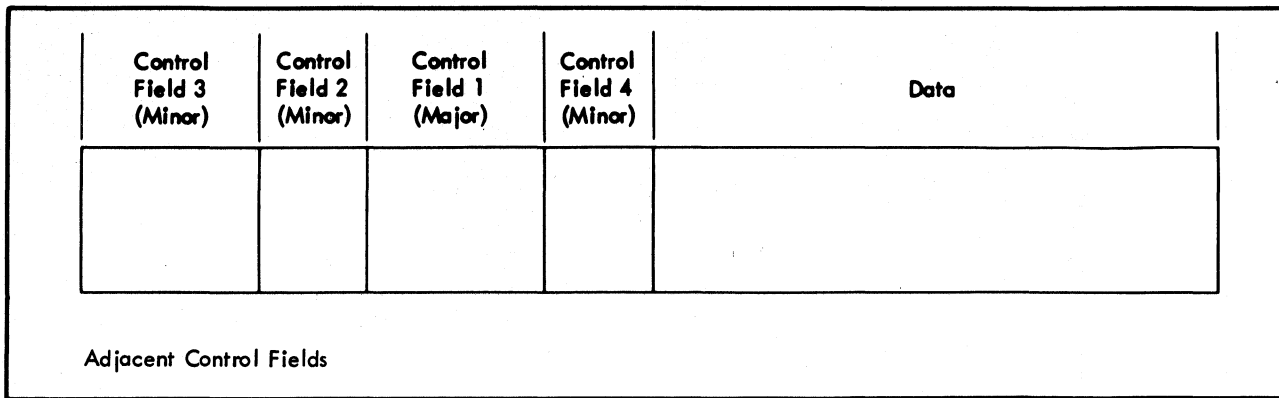


Figure 7. Control Field Formats

A given control field must be located in the same relative position in each record of all files. If, for example, the first two bytes of the records in a file are designated as the major control field, the program assumes that these two bytes are the major control field in every record of that file.

If a KEY is associated with each disk input record (fixed-length unblocked records only), the KEYLEN entry can be

specified in the OPTION statement, and the records can be sorted on the contents of the KEY field.

Control data may be in any one of the following formats:

1. Binary (character).
2. Packed- or zoned-decimal.

3. Normalized floating point (in either the short or long format).
4. Fixed-point.

The following limitations must be observed when setting up the control fields for a file:

1. A control field for a variable-length record cannot include the record-length field.
2. Each control field must begin and end on a byte boundary.
3. Each control field must be at least one byte long.
4. Each control field may be a maximum of:
 - 256 bytes when using binary (character) data.
 - 16 bytes (including the sign) when using decimal data.
 - 4 bytes when using normalized floating-point data (short format).
 - 8 bytes when using normalized floating-point data (long format).
 - 256 bytes (including the sign) when using fixed-point data.
5. The total of the control-field lengths must not exceed the input record length (the minimum record length minus four when processing variable-length records), unless overlapping control fields are used.
6. When using either zoned-decimal data or packed-decimal data, the signed fields must be either all ASCII or all EBCDIC; they must not contain a combination of both.

Files may be sorted or merged in either ascending or descending sequence. The sequence of each control-data field is independent of the other control-data fields. For example, a user may specify ascending sequence for the major control fields, and descending sequence for all the minor control fields.

The control-data fields are defined for the program by means of user-prepared control statements. These statements specify the type of operation (sort or merge), the sequence (ascending or descending), and the size and location of each control-data field.

For further details about data formats, see IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.

DATA CONVERSION

The sort/merge program includes data-conversion and -reconversion routines. Data in the format of floating point, fixed point, packed decimal, or zoned is converted, before processing, to binary so that it can be compared. The data is reconverted to its original form before the output file is written.

USER MODIFICATION

Phases 1, 3, and 4 of the sort/merge program provide for the addition of user-written routines. Linkage between the program and these routines is provided by branch-and-link instructions (BAL) called exits. These exits are:

Phase	Exit	Use	Return Branch
1	11	Check nonstandard input labels (tape only) or user input labels (tape or disk).	B 0(14)
1	12	Translate or modify input records.	B 0(14)
1	13	Process incorrectly read records (tape only). Bypass incorrectly read records (tape only).	B 0(14) B 4(14)
3	31	Create and write nonstandard output labels (tape only). Create user output labels (tape or disk). After last user label	B 0(14) B 0(14) B 4(14)
3	32	Alter or delete records, translate data, summarize records, shorten or lengthen records. Insert records. After last insertion	B 0(14) B 4(14) B 0(14)
4	41	Similar to exit 11, but includes mixed labels	B 0(14)
4	42	Convert data.	B 0(14)
4	43	Same use as exit 32.	B 0(14) or B 4(14)
4	44	Same use as exit 31	B 0(14) or B 4(14)
4	45	Similar to exit 13, but includes tape <u>and</u> disk	B 0(14) or B 4(14)

For a detailed description of user exits and user routine formats, see IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.

CHECKPOINT, INTERRUPT, AND RESTART

The checkpoint, interrupt, and restart feature permits the user to interrupt the sorting process, and conveniently restart

it, at the end of phase 1 and during phases 2 and 3. If a job is interrupted during the assignment phase or phase 1, the job must be restarted from the beginning. The feature will not function in a merge-only operation.

The sort program automatically writes a checkpoint record on disk during the assignment phase and updates it at the end of phase 1 and at the beginning of each pass in phases 2 and 3. The checkpoint record is composed of information required to restore the program to the beginning of the interrupted pass (the end of phase 1 is considered to be the start of phase 2).

Execution of a restart restores the contents of the section of main storage recorded at the time of checkpoint, and processing is continued from the beginning of the interrupted merge pass.

CHECKING FEATURES

Two sort/merge checking features are included in the program: A record-count feature and a sequence-check feature.

The record-count printout is automatic. A count of the records processed will be printed on SYSLOG at the end of phases 1, 3, and 4. In phases 3 and 4, the count will not include user insertions or deletions.

A sequence check occurs automatically during phases 3 and 4. If a sequence error is detected, the system prints a message and awaits the operator's instruction to continue processing or to cancel the job.

LABEL PROCESSING

The sort/merge program processes IBM System/360 standard header and trailer labels as part of its input/output processing. Standard file labels are mandatory for disk input/output files; non-standard or no labels are permitted only with tape files.

Programmed exits are provided to read and check nonstandard tape-input headers and trailers, and to compile and write non-standard tape-output headers and trailers. If the user specifies nonstandard tape input labels, but does not specify the accompanying exits, the program will search for the tape mark that separates the labels and the data, thus bypassing the nonstandard labels.

Unlabeled tape files can be processed by the program. The first record read from an unlabeled input volume can be either a tape mark or a data record. If it is not a tape mark, the program will backspace the tape to the beginning of the first record.

A tape mark can be written before the first data record, and one will be written after the last data record on each output volume with nonstandard labels or no labels.

For a complete discussion of label processing, see IBM System/360 Disk Operating System, Data Management Concepts, Form C24-3427, and IBM System/360 Disk Operating System, Supervisor and I/O Macros, Form C24-3429.

SYMBOLIC UNIT ASSIGNMENTS

Assignment of input/output devices to specific channels, if a 2-channel system is used, is left to the discretion of the user. The sort/merge program does not require a particular device on a particular channel. The program-required disk work area can be composed of a maximum of six individual extents. The total number of tracks required for a work area (the sum of the extents) must not be greater than the total number of usable tracks on six disk packs.

Input to a sort or a merge-only operation can be read from multiple disk drives, and output can be written on multiple disk drives. The input and output areas are flexible; multiple extents can be specified for each input file and for the output file. These I/O areas can be arranged around the program-required work area, which must be on-line at all times during execution. Regardless of the arrangement used, the program maximum of eight drives must not be exceeded. The user is free to assign any symbolic unit name to the disk drives used for sorting or merging, providing the limitations established by the disk-resident control program are observed.

When the system includes tape drives, primary and alternate tape drives can be specified to eliminate the time required to rewind and unload a tape reel and mount a new reel either when reading a multivolume input file or when writing a multivolume output file. For example, if an input file is contained on two volumes, the second volume can be made ready on the alternate drive while reading from the primary drive. This facility exists for both sort and merge-only operations.

The program assumes, if tape input is specified, that the initial reel of the first input file to be sorted or merged will be read from the tape unit with the symbolic address SYS002, and that the alternate input unit will be assigned as the alternate unit for SYS002. In the same manner, the program assumes that the initial volume of the second input file, for either a sort or merge-only run, will be read from the unit assigned to SYS003; the alternate unit will be assigned as the alternate for SYS003. If tape output is specified, the program assumes the output file will be written on the unit with the symbolic address SYS001. The alternate unit, if any, is assigned as the alternate for SYS001.

For a sort operation, the tape units with the symbolic addresses SYS002 through SYS010 are assumed to contain the initial volumes of input files 1 through 9, respectively. For example, if two tape files are to be sorted, the initial or only volumes must reside on SYS002 and SYS003, which can be assigned to the same tape drive or to different drives.

For a merge-only operation, the tape units with the symbolic addresses SYS002 through SYS005 are assumed to contain input files 1 through 4, respectively. For example, if three tape files are to be merged, the initial or only volumes must be mounted on SYS002, SYS003, and SYS004, which must be assigned to different drives (initial volumes of all files must be on-line).

Figure 8 illustrates one example of an input/output scheme for a sort or merge-only operation. In this example, two files can be sorted or merged with a minimum of operator intervention by utilizing alternate drives. The system illustrated uses six tape units, a central processing unit, and one or more disk drives. The diagram is explained as follows:

1. Two input files, FILEA and FILEB, are to be sorted or merged.
2. Each file consists of two volumes.
3. The initial volumes of FILEA and FILEB are mounted, as required, on SYS002 and SYS003, assigned respectively to tape units A and C.
4. The alternates assigned to SYS002 and SYS003 (via ASSGN statements) are tape units B and D, respectively.
5. SYS001, the symbolic unit address for FILEO, is assigned to tape unit E with an alternate unit F.

6. The output file consists of three volumes. The first is to be written on unit E, the second on unit F, and the third on unit E.

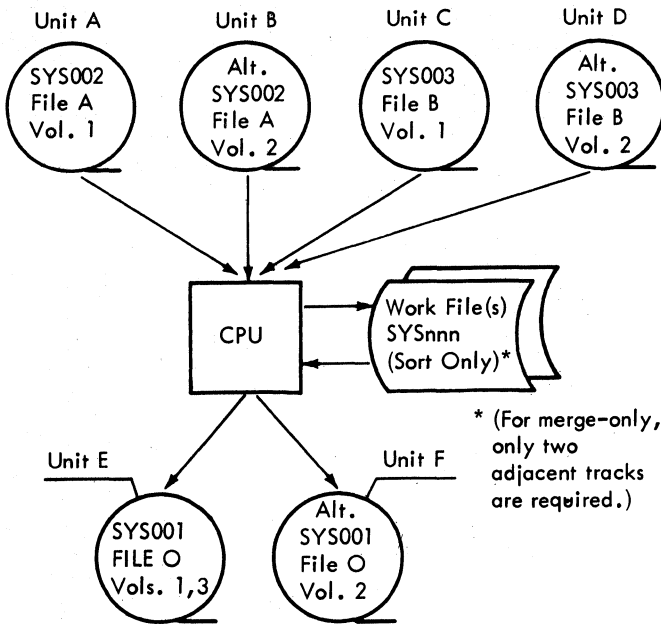


Figure 8. Tape Input/Output Scheme for Sort or Merge-Only Operation

SYSTEM REQUIREMENTS

The System/360 used to run the sort/merge program must have at least:

- 16K (16,384) bytes of main storage.
- Standard instruction set.
- One IBM 2311 Disk Storage Drive attached to either one multiplexor channel or one selector channel. (System residence may require the user to have an additional disk storage drive for sorting.)

- One IBM 1403 or 1443 Printer; or one IBM 1052 Printer-Keyboard
- One IBM 1442, 2520, or 2540 Card Read Punch, IBM 2501 Card Reader, or IBM 2400-Series Magnetic Tape Unit (7- or 9-track) assigned to SYSIPT and SYSRDR.
- One IBM 2400-Series Magnetic Tape Unit, if tape input/output is desired.

The sort/merge program will also operate with:

- A maximum of 512K (524,288) bytes of main storage.
- A maximum of eight IBM 2311 Disk Storage Drives (including the system residence drive), two to six of which can be used for intermediate storage.
- One to ten IBM 2400-Series Magnetic Tape Units (7- or 9-track) for input/output to a sort operation. Any number of alternate tape units can be supported.
- One to five IBM 2400-Series Magnetic Tape Units (7- or 9-track) for input/output to a merge operation. Any number of alternate tape units can be supported.

The possible combinations of 7-track and 9-track tapes for the sort operation are:

<u>Input</u>	<u>Output</u>
7	7 or 9
9	7 or 9

The possible combinations of 7-track and 9-track tapes for the merge-only operation are:

<u>Input</u>	<u>Output</u>
7 and/or 9	7 or 9

The assignment phase reads, from SYSIPT, the sort/merge control cards prepared by the user. The control card statements contain file description and necessary parameters to perform a specific sort/merge operation.

The sort/merge control cards need not be loaded into the reader in any specific sequence except that the END control card must follow the control deck. The control information is rearranged in main storage, in accordance with the control statement sequence code, so that the SORT (or MERGE) statement is processed before any of the other control statements.

As the statements are processed, tables are built and areas are defined for use in succeeding phases. The constants that are calculated for a sort/merge run include:

- M - order of merge
- G - size of internal sort sequence
- B - sort blocking factor

Internal diagnostics check the format of the control statements and the values within the tables are cross-checked. Values are assumed for certain optional and required parameters that may have been omitted from the control statements; for example:

- in the SORT statement, an ascending order is assumed for a missing sequence parameter (Sn). The assumed value is flagged as an error, but the assumption allows processing to continue to the end of the phase.

- when processing fixed-length records, the field definer (LENGTH) in the RECORD control card need only contain an entry for L1 and not for L2 and L3. This is a valid condition and does not indicate an error.

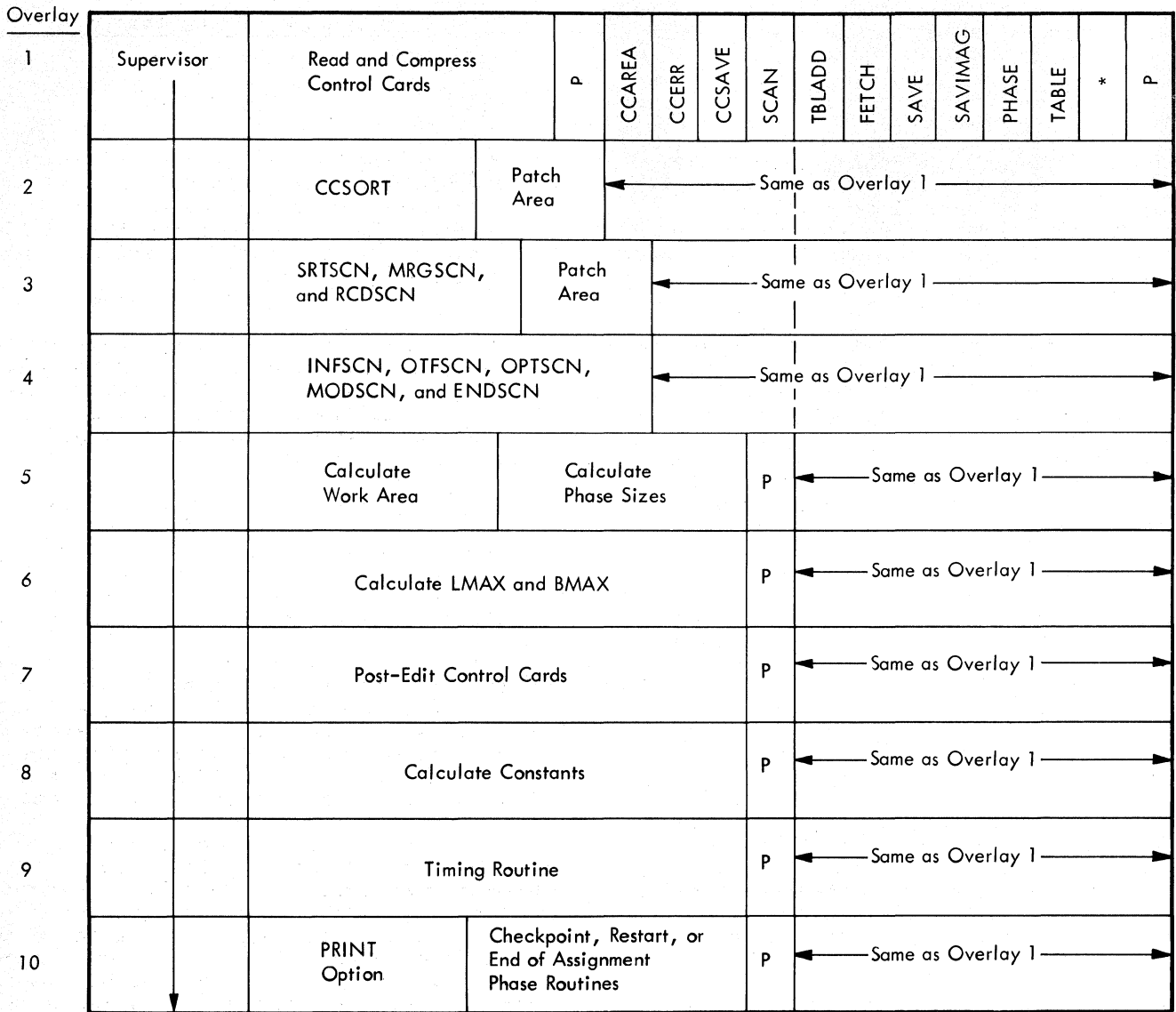
- If FILES (an optional parameter) is omitted, its value is assumed to be 1. Thus, if FILES is omitted, the volume, input, and label entries must be compatible with FILES = 1 to avoid subsequent errors.

The user can request that the actual control cards as well as a detailed description of the given parameters be printed on SYSIST. To accomplish this optional program feature, the user must specify the PRINT option.

Messages are printed defining the error conditions detected by the assignment phase. If SYSLOG is a 1052, the user operator has the option to either cancel the job or to retry after correcting the control card errors. If SYSLOG is not a 1052, the job is automatically canceled.

When a RESTART run is to be made (RESTART runs are not available for merge-only operations), phase 2 is fetched at this time. Otherwise, constants for either a sort or a merge are written in the 2311 disk checkpoint area (last four tracks of the given FILEW area), and either phase 1 (for a sort) or phase 4 (for a merge) is fetched.

Figure 9 is a layout of main storage for the assignment phase, showing each overlay and its respective routines.



Note: Not drawn to scale.

* - { ABORT INITWTR
 OPERROR DTF [Read Control Cards
 ERROR Print Messages
 RDRTN CPMOD

P - Patch Area

Figure 9. Assignment Phase Main-Storage Layout

READ AND COMPRESS CONTROL CARDS - AA

This routine (overlay 1) is the first overlay of the assignment phase; it is used to read control cards into storage, check their validity, and prepare any error messages that may be required. Phase tables and other areas used throughout the assignment phase are contained in this overlay.

Control cards are read one at a time into a read-in area and checked for:

- User run code
- Blank in first column
- Blank control card
- Valid statement definers
- Duplicate definers
- Definer in correct card position
- Valid continuation cards
- PRINT, CALCAREA, and ADDRROUT options
- END card after reading 25 control cards.

Valid control cards (including their continuation cards, if any) are processed to include a code at the beginning of the card image (to denote SORT, INPFIL, etc.), and compressed to eliminate blank columns. A slash (/) is placed at the end of the card image which is then moved to a save area for further processing in subsequent overlays.

The OPTION card is checked for PRINT, CALCAREA, and ADDRROUT in this routine and the selected bit is set in the phase table.

Control cards are processed until either an END card or the twenty-fifth control card has been read. In the former case, overlay 2 is fetched; in the latter case, if an END card has not been read, an error message is printed.

Other error messages are prepared, depending on the type of error (if any), but are not printed at this time.

This overlay includes some small subroutines that will remain in main storage and will be used by subsequent overlays:

- Read 1052 replies
- Write

- Print error messages
- Print control card contents
- Abort

The print and abort subroutines are initialized for the printing devices and for operator response depending on the available system features and I/O assignments.

DSORT, AA-B1

The three base registers to be used in this overlay are initialized.

The end of address of the supervisor area is calculated and stored. A series of tests is then made to determine if SYSLOG is a printer and, if so, if it is designated for the same device as is SYSLST. Switches and indicators are set, depending on the result:

- SYSLOG is not a printer - turn on bit #7 in SYSLGBIT to indicate SYSLOG is an IBM 1052 Printer-Keybaord
- SYSLOG is a printer (1403 or 1443) but is not equal to SYSLST - activate branch at SW140343 to use the printer as SYSLOG and activate branch at ABORT to bypass operator response routine.
- SYSLOG is the same printer as SYSLST - turn on bit #6 in SYSLGBIT to indicate SYSLOG = SYSLST, activate branch at OPERRSW to print only on SYSLST, and activate branch at ABORT to bypass operator response routine.

Other information is then retrieved from the communications region and, if necessary, the current date is converted to MMDDYY format.

The starting and ending address of the message save area (CCSAVE and CCSAVE + 735) and the starting address of the card image save area (SAVIMAG) are stored in CCSADD, CCEND, and CIMAGSV, respectively.

BEGIN1, AA-C1

Three registers are initialized for use during the scanning of control cards:

- R7 - Index for control card area
- R3 - Marker for read-in area address

• R4 - Pointer to end of scan area

TSTBLKCD, AA-J2

A test is then made to determine if 25 control cards have been read. If such is the case at this point, an error condition exists because no END card has been read; a branch is made to ERR4. If the card count is less than 25, a branch-and-link is then made to RDRTN to read a control card into location READIN. The routine then continues to RUNCODSW.

As long as blank columns are detected (from column 2 on), the routine continues testing each succeeding column for a blank. If the end of the card is reached, indicating a blank card, a branch is made to ERR6. As soon as a column that is not blank is detected, the card image is moved into the save area (from READIN to CIMAGSV) and the routine continues to SW.

ERR4, AA-C4

Information for the error message at location E04 is obtained, and a branch-and-link is made to OPERROR to print the message. A branch is then made to LOAD1.

ERR6, AA-K2

A bit is set in indicator E06BKT so that message 7D061 will be printed at the end of overlay 4. A branch is then made to BEGIN1+4.

RUNCODSW, AA-G1

This location is initially a no-op and it remains so until a run code is given by the user (this code is found in columns 73-80 of the sort/merge statements and is used by this phase as an identifier). The code is stored at RUNCOD and this no-op is made an unconditional branch to COUNTCDs. Until such time, the routine tests each card for a run code before continuing to COUNTCDs.

SW, AA-B2

This location is initially a no-op and is made a branch to COL16 only when a continuation card is to be processed, as determined during the scan of the preceding card. When SW is a no-op, a branch is made to CKPARA.

At COUNTCDs, the control card counter (CDCOUNT) is increased by one and the routine continues to READB to test for a blank in the first column. If the first column is blank (as it should be), the routine continues to TSTBLKCD; if not blank, a branch is made to ERR1.

COL16, AA-C2

If the scan register has not reached column 16, the columns preceding column 16 are tested for blanks. If a non-blank column is detected before reaching column 16, the card is in error; the branch at SW1 is activated (to permit this card to be processed) and a branch is made to ERR5.

ERR1, AA-J1

The control card number is converted to decimal and inserted in the error message at E01. The message is then moved to the message save area (CCSAVE) if space permits; if not, the message is not moved and a branch is made back to BEGIN1+4 for the next card. If the message is moved to CCSAVE, a branch is made to BEGIN1+4 except when the card just read is an END card, in which case the branch is to LOAD1.

When the scan has reached column 16 and the column is blank, the card is in error and a branch is made to ERR5. If column 16 is not blank, the card is valid and a branch is made to UPMOVE to process it.

ERR5, AA-E2

The control card number is converted to decimal and inserted in the error message at E05. The message is then moved to the message save area (CCSAVE) if space permits. (If the save area is full, the message is not moved and the routine continues.) The branches at SW1 and BSLASH are made no-op's (because it is not yet known if the next card is a continuation card) and the routine continues to SW1.

SW1, AA-F2

This branch is activated when the information on a continuation card begins before column 16. In these cases, the branch is made to UPMOVE to try to process the card. Otherwise, this location is a no-op and a branch is made to BEGIN1+4 to read the next card.

CKPARA, AA-A3

The statement definer code is determined by testing for each definer (SORT, MERGE, etc.) and when one is found:

1. Location DUPBK is tested for a duplication of the definer (it may have appeared on a previous card). If such is the case, a branch is made to ERR64.
2. If the definer is not a duplicate, the appropriate symbol is placed in DUPBK (for future testing of duplicate definers) and the definer length is placed in location SUBTRC.
3. The address of the instruction for inserting the appropriate definer code is placed in register R9; a branch is then made to INITST.

The labels and constants used for the various definers are:

<u>Definer</u>	<u>DUPBK</u> <u>Symbol</u>	<u>Byte</u>	<u>Length in</u> <u>SUBTRC</u>	<u>Address in</u> <u>Register R9</u>	<u>Code to be</u> <u>Inserted</u>
SORT	S	0	5	INITCS	1
MERGE	M	1	6	INITCM	2
RECORD	R	2	7	INITCR	3
INPFIL	I	3	7	INITCI	4
OUTFIL	O	4	7	INITCO	5
MODS	E	5	5	INITCE	6
OPTION	O	6	7	INITCP	7
END	D	7	4	INITCD	8

If none of the listed definers are found, a test is made to determine if 25 cards have been read and if so, a branch is made to ERR4; if not, the branch is to ERR3.

ERR64, AA-C3

The duplicate statement definer is moved into the error message at E64. The message is then moved to the message save area (CCSAD) if space permits, and a branch is made back to BEGIN1+4 for the next card. (If the save area is full, the message is not moved.)

INITST, AA-A5

A test is made to determine if the statement definer is in the correct position on the card (between columns 2 and 15). If not, a return switch (ERR3A) is set and branch is made to ERR3; upon return, the ERR3A switch is turned off and a branch is made to the address in register R9 (see list at end of text under CKPARA).

When the definer position is valid, a branch is made directly to the address in register R9. At these locations, the definer code that was found is placed in the byte immediately to the left of the control card image in the save area

(CCAREA) and then, depending on the definer, one of three branches is made:

- OPTION definer - to OPTSC1
- END definer - to LOAD1
- All others - to UPMOVE

ERR3, AA-B5

The invalid statement definer is moved into the error message at E03. The message is then moved to the message save area (CCSAD), if space permits, and the routine continues to ERR3A. (If the save area is full, the message is not moved.)

ERR3A, AA-C5

This location is a return switch that is turned on (unconditional branch) when ERR3 is to be entered from INITST. If such is the case, a branch is made from this point back to that portion of the routine (see first paragraph under INITST).

When ERR3 is entered from CKPARA, this switch is off (no-op) and a branch is made to BEGIN1+4 for the next card except when the card just read is an END card, in which case the branch is to LOAD1.

OPTSC1, AA-F3

The switch at location OPTSCS is turned on (unconditional branch to OPTSC2) and the address of the first byte of the current control card is stored. A branch is then made to UPMOVE.

UPMOVE, AA-G3

The continuation-card switches at SW and SW1 are turned off (made no-ops) and, depending on the results of several tests, the appropriate course of action is taken:

- Continuation column blank but column 71 not blank - Branch to BYTE1, where the end-of-scan position is incremented by one to determine the slash (/) position that denotes the end of the card image. The slash is then moved so that it is immediately to the right of the high-order byte of the card image. The

slash position is saved and the compressed card (including the slash) is moved to the control card area (CCAREA).

- Continuation column blank and column 71 blank - Scan from column 71 back, decrementing the end-of-scan position each time, until the first valid byte is found. Then, branch to BYTE1 and process as described in the preceding paragraph.
- Continuation column not blank and column 71 not blank - Turn on switch SW and branch to BYTE, where the slash (/) position is saved and the compressed card (including the slash) is moved to the control card area (CCAREA).
- Continuation column not blank but column 71 blank - Turn on switch SW and test if current card is valid; if not, branch to ERR5. If card is valid, scan from column 71 back, decrementing the end-of-scan position each time, until the first valid byte is found. Then, branch to BYTE and process as described in the preceding paragraph.

At this point, the current card, if valid, has been compressed so that no blanks will be included when the card image is moved, except for the blank between the statement definer and the first field definer.

If the next card is to be a continuation card, the switch at BSLASH is a no-op and the routine initializes so that the information from the next card will be moved to CCAREA as a continuation of that from the present card. A branch is then made to BEGIN1+4 to read in the next card.

When it has been determined that the next card is not a continuation card, the scan area is updated and the routine continues to OPTSCS.

OPTSCS, AA-J3

This location is a no-op for all cards except the "OPTION" card and, in all such cases, a branch is made back to BEGIN1+4 to read in the next card. However, when the current card is the "OPTION" card, this location would have been made an unconditional branch (see OPTSC1) and the routine now branches to OPTSC2.

OPTSC2, AA-K3

The switch at OPTSCS is turned off and tests are made for the PRINT, CALCAREA, and ADDRROUT options. The corresponding bit in table TAGTYPE is set for each option specified, and a branch is made back to BEGIN1+4 to read in the next card.

LOAD1, AA-E5

The control card image area (CIMAGSV) is closed by inserting a hexadecimal "EF" at the end of the last card image in the area. The current message save area address (CCSADD) is then checked to determine if it is at the upper limit (CCEND). If CCSADD is less than CCEND, a hexadecimal "FF" is inserted at the end of the last message in the area; if CCSADD is equal to or greater than CCEND, the FF indicator is not required. A branch is then made to FETCH.

FETCH, AA-F5

This location contains an expansion of the FETCH macro; it is used to fetch the next required overlay as determined at different points throughout the assignment phase overlays 1 through 9. The number of the overlay to be fetched (2 through 10) is inserted in the operand for the macro before branching to this location.

At the end of overlay 1, therefore, overlay 2 (Chart AB) is fetched for execution.

CCERR, AA-G4

This location is entered from overlays 2, 3, and 4 of the assignment phase and is used to print the entire compressed contents of a particular control card only when the first error in that card is detected. This is done so that if the error indication was caused by a keypunch error, it may be corrected by the operator before canceling the job. Subsequent errors in the same card will cause only an error message to be printed, but not the entire card.

The term "card" in the preceding paragraph means a logical card, which includes continuation cards, if any, for a statement. Provision is made for printing longer logical cards 80 bytes at a time.

After the contents of a particular card are printed, a branch is made to OPERROR to print the first error message. Subsequent entries for the same card branch directly to OPERROR.

OPERROR, AA-J4

This location is entered from ERR4 and ABORT in this overlay and from several points in other overlays (either directly or through CCERR). The entries are made via a branch-and-link with link register RA.

If SYSLOG and SYSLST are assigned to the same printer, a branch is made back to the address in link register RA where a branch-and-link to ERROR will be found; messages will thus be printed on SYSLST only. However, if SYSLOG is not the same printer as SYSLST, a branch is made to PRT1403 and the messages are printed on both SYSLOG and SYSLST.

If SYSLOG is an IBM 1052 Printer-Keyboard, the message is typed out and a branch is made back to the address in link register RA where a branch-and-link to ERROR will be found. The messages are then printed on the SYSLST printer.

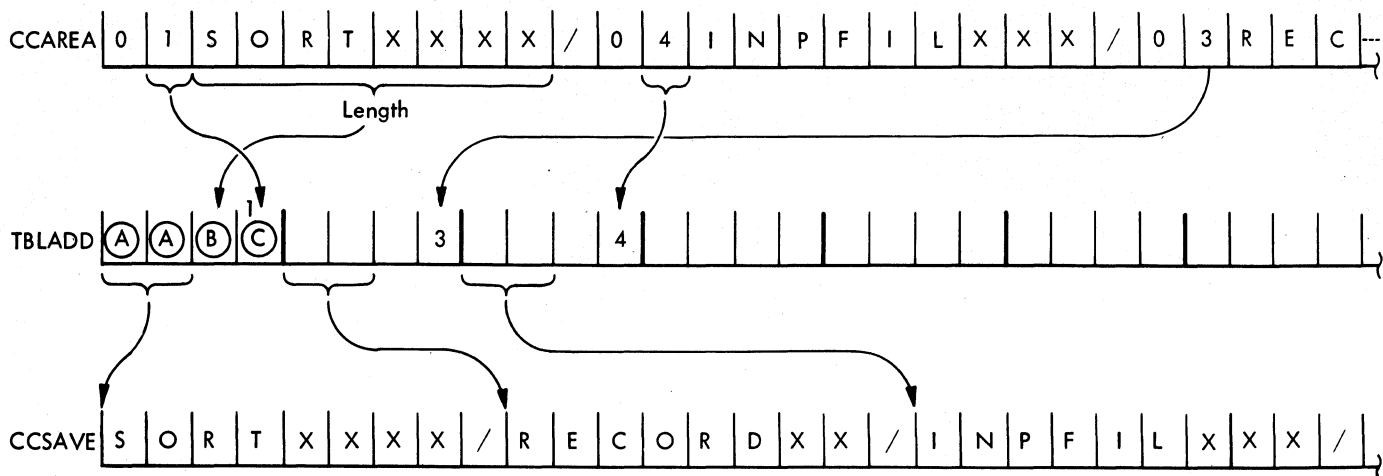
In all cases, after the messages are printed the program returns to the address in the link register RA to continue processing.

ABORT, AA-H5

This location was initialized at the start of overlay 1 to permit operator response for REPLY or CANCEL only if SYSLOG is a 1052 and if SYSIPT and SYSRDR are not tapes.

ABORT is entered from overlay 4 (Chart AE) when there is an error of a type that will not permit the assignment phase to continue processing beyond that point. If operator response is permitted, the message at location E90 is printed via a branch-and-link to OPERROR. The operator may then cancel the job or correct the error and retry. In the former case, a CANCEL macro is issued; in the latter, the program returns to the start of overlay 1 after the job cards and sort cards have been placed in SYSRDR and SYSIPT, respectively.

If a retry is not permitted because of the restrictions mentioned, the message at E92 is printed and the job is canceled.



AA - 2-byte address of the compressed control card in CCSAVE.
 B - total number of bytes (excluding code) in compressed control card.
 C - code indicating type of card.

- 1 - SORT 5 - OUTFIL
- 2 - MERGE 6 - MODS
- 3 - RECORD 7 - OPTION
- 4 - INPFIL 8 - END

Figure 10. Sort Compressed Control Cards

SORT COMPRESSED CONTROL CARDS - AB

This routine (overlay 2) initializes TBLADD and CCSAVE so that each logical control card can be found and scanned by its respective overlay for errors and extraction of information. The order in which the codes are placed in the table (TBLADD) and in the area (CCSAVE) is: 1-SORT, 2-MERGE, 3-RECORD, 4-INPFIL, 5-OUTFIL, 6-MODS, 7-OPTION, 8-END. Refer to Figure 10.

there is no error, control is passed to MOVECC. Control cards are checked for invalid codes and missing mandatory codes. Overlay 3 is fetched when the END card code is reached.

MOVECC, AB-D3

DSORT002, AB-B1

The base register is initialized and the stacked message from overlay 1 is printed. The routine then continues to CCS0.

If a code is found, the respective area in the table at location TBLADD is updated with the code and length of the logical card. The card is moved to the area labeled CCSAVE, and its starting address is placed in TBLADD. The code search register is incremented to search for the next sequential respective code in CCAREA. Control is returned to the CCS0 routine at label CCS2, until all cards have been scanned.

CCS0, AB-D2

CCAREA is scanned until a 1 code is reached, then a 2 code, and so on. When a valid code is read in its proper order and

SCAN SORT, MERGE, AND RECORD CONTROL CARDS
- AC

SRT1, AC-B3

This routine (overlay 3) scans the compressed control cards for the statement definer of each card. As the statement definer is read, the appropriate overlay for that card is determined. The statement definers and their overlays are:

- SORT, MERGE, and RECORD -- overlay 3
- INPFIL, OUTFIL, MODS, OPTION, and END -- overlay 4

If it is determined that a SORT or MERGE card is present, overlay 3 continues and scans the control card for values and errors. The values are placed in the phase tables starting at the label FORMAT. Insertions are made for missing parameters (either required or optional) and errors are explained by messages. When the card has been completely scanned, control is returned to the initial scan routine of overlay 1, which determines the next overlay to be fetched.

When the appropriate overlay has been fetched, the next control card is scanned. Overlays 3 and 4 do not overlay the initial scan routine of overlay 1.

After all control cards have been scanned, no errors detected, and the table built, overlay 5 is fetched.

DSORT003, AC-B2

Registers are initialized to scan the CCSAVE area for a control card and to call in the corresponding overlay. The routine then continues at SRTSCN.

SRTSCN, AC-C2

The address of the first sorted control card is retrieved from TBLADD and stored in MSGADD. A test is made to determine if the card designates a sort or a merge. If a merge, a branch is made to MRGSCN; if a sort, the sort indicator (S-character) is set in MANDBK, and the index register is initialized with the control field length minus one (CF1LNG). A test is made to determine that the proper format for the control card has been used. If an error is detected, a branch is made to ERR1 to execute the corresponding error routine. If no error is detected, the routine continues at SRT1 to scan the control card.

Scanning of the control card is continued one byte at a time until a complete statement, SORT or MERGE, has been scanned (a slash followed by a hex 61 indicates end of a statement). A comparison is made between the field definer and the dictionary of field definers located at CKFLDS. When a valid definer is found, the program branches to the respective routine, which extracts the values for that definer and places them in the phase table. The routines are:

- FLD - field definer
- FILRTN - order of merge for merge-only; or number of files for sort
- FMAT - format definer
- FILSZE - size definer

If a valid field definer is not found, the program branches to ERR2 to print the appropriate error message.

FLD: The FIELD definer is bypassed, and a check is made for the left delimiter. The values for each control field are extracted, converted to useable form, and placed in the phase table. For example, if there is only one control field, the code placed in the phase table is 0080 if ascending, or 00E0 if descending sequence. However, if there is more than one control field, the code for ascending sequence in the first field is 0000, and 0080 in the last field, if it is designated ascending. For descending sequence, the first field is 0060, and the last field, if descending, is 00E0. Further checks are made for the position and the length of a field. A missing sequence value is assumed to be ascending and an error message is printed to indicate that the sequence value was not given.

A maximum of 12 control fields can be specified with this format. The total length of all control fields must not exceed 256 bytes.

FILRTN: The number of user-given files is extracted from the control card and converted to binary. The binary number is stored at FILES and, in addition, is used as the order-of-merge for a merge-only run. If FILES is left out of the control card, the program assumes the number of files to be 1 for either a sort or a merge.

Note: Errors may be encountered later in the phase and messages will be printed if FILES is not given and assumed to be 1. For example:

FILES GIVEN EQUAL TO 1, AND INPUT
EQUALS T,D,T

If a sort, no errors would be detected because the input could then be only disk or tape, but not a combination of the two.

FMAT: The field definer is bypassed, the type of format is determined, and the appropriate bit is set ON in the phase table at FORMAT. The user's original code is saved for use with the PRINT option at FMATBK, if specified. The format codes are:

- FL - floating point
- BI - binary
- FI - fixed point
- PD - packed decimal
- ZD - zoned decimal

FILSIZE: The field definer is bypassed and the user-given file size is stored in the phase table at FILESZ. Size is required for a sort, but not used for a merge. The sort operates at greater efficiency even if the size entry is only an estimate; however, the estimated size entry must be equal to or greater than the actual file size. Otherwise, the work area calculated would be incorrect and an error message will be printed.

When the end of the control card is reached, control is returned to this routine at SCAN.

SCAN, AC-C4

If the card is a MERGE or RECORD control card, a branch is made to MRGSCN or RCDSCN, respectively, in this overlay. For control cards other than SORT, MERGE or RECORD, overlay 4 is fetched.

If, for example, an OUTFIL card is detected in CCSAVE, the overlay number (byte 19) is checked to assure that the required overlay is in main storage. Because both INPFIL and OUTFIL use the same overlay, overlay 4 will be in main storage when the OUTFIL card is detected.

MRGSCN, AC-D2

Registers are loaded with the MERGE control card location in CFILNG and TBLADD, and the card code is tested. If there is a MERGE

card in TBLADD, the merge indicator (M-character) is placed in MANDBK+1 and a branch is made to SRT1. If there is not a MERGE card, the branch is to RCDSCN. An error is detected if both merge and sort are specified for the same run or if neither is specified. In either of these cases of error, control is given to the operator (if SYSLOG is a 1052) or the job is terminated.

RCDSCN, AC-F2

General registers, the index register, and the return switch are initialized and the error switch is reset to OFF. The routine then continues at COMP.

COMP, AC-G2

Control cards are scanned until RECORD card code is found. The RECORD indicator (R-character) is placed in MANDBK+2. The address of the RECORD control card (in TABLADD) is stored in MSGADD and a branch is made to RC1.

RC1, AC-H2

The field definer is located and a branch is made to the respective routine:

- CKTYPE -- to scan the type definer for F or B (fixed- or variable-length records).
- CKLENG -- to scan the "length" definer.

When a comma follows the first field definer set of values, the next definer will be processed. A branch is made back to BEGIN2 to scan the next control card. If errors are detected the program branches to the CCERR routine which is described in overlay 1.

CKTYPE: This routine turns on the appropriate bit in the phase table for fixed- or variable-length records.

RECTYP: Fixed-length records are assumed if no value is found for record type, but an error condition results. If blocking is variable, the assumption is changed to variable-length in a later overlay and an error indication is again given.

CKLENG: This routine stores the I/O record lengths in the phase table at INTPRL. If only the L1 value is given, a branch is made to the routine beginning at label ASSUM1 to assume a value for L3 (for fixed-length records) or for L3, L4 and L5 (for variable-length records). This function is not applicable to ADDROUT or merge runs. L2 is not used by the sort/merge program. An error condition results if no value is given for L1 or if the wrong value assumption is forced for L3 thru L5.

Control is returned to SCAN when the end of a card is detected.

SCAN INPFIL, OUTFIL, MODS, OPTION AND END CONTROL CARDS - AD, AE

Overlay 4 is fetched to scan:

- INPFIL card
- OUTFIL card
- MODS card
- OPTION card
- END card

DSORT004, AD-B1

Registers are initialized and the error switch is reset to OFF.

INFSCN, AD-C1

The INPFIL indicator (I-character) is placed in MANDBK+3 and MSGADD is updated with the address of the INPFIL card (from TBLADD) for use in case of error conditions. A branch is then made to INF1.

INF1, AD-G1

This routine is initialized to accept either an INPFIL or OUTFIL control card. The program locates the field definer and branches to the respective routine. For an INPFIL card, the routines are:

- BLOCKI -- block size
- XSCAN -- block type

- VOL1 -- volume
- CKINPT -- input media
- CKOPEN -- I/O open file routine
- CLIFLE -- I/O close file routine
- BYPASS -- bypass unreadable records (tape input for sort, tape or disk input for merge)

BLOCKI: A check is made to determine whether an INPFIL or OUTFIL card is present. If it is an INPFIL card and it has no delimiters, variable blocking is assumed. The input block size value is placed in the phase table at BLKSIZ.

XSCAN: The value (X) for fixed-length blocks is extracted and an indicator is initialized at BLKSIZ+3. The program assumes variable-length blocks if there is no second value in BLKSIZE. The appropriate bit is turned on in BLKSIZ+3, and an information message is given to the operator if the print option is specified.

VOL1: Scans for volume value (tape only). For a sort run, up to nine volume entries are allowed and are placed in bytes 0 through 8 of MRGVOL. Any entries over the maximum will cause an error condition. For merge-only, a maximum of four entries is allowed: FILEA, FILEB, FILEC, and FILED (bytes 0 through 3 respectively).

CKINPT: The input media specified by the user is extracted and the appropriate bit is set in INPUTMRG+1 for a sort, or in INPUTMRG+1 through INPUTMRG+4 for a merge.

CKOPEN: The tape-file open options are extracted and the appropriate bit for input is set in BLKSIZ+3.

CLIFLE: The tape-file-input CLOSE options are extracted and the appropriate bit is set in BLKSIZ+3.

If any errors are detected, control is passed to CCERR in overlay 1. At the end of the card, control is returned to SCAN.

For a OUTFIL card, INF1 is initialized to handle output file card routines:

- BLOCKO -- output block size
- COPENO -- output open option
- CLOFILE -- output close option
- CKOUT -- output device
- CKNOTPMK -- NOTPMK option (tape output)

BLOCKO: The BLOCKI routine is initialized to extract output block size and place it, in binary form, in BLKSOZ.

COPENO: The tape-file-output options are extracted and the appropriate bit is set in RECTYP+1.

CLOFLE: The tape-file-output CLOSE options are extracted and the appropriate bit is set in RECTYP+1.

CKOUT: The output media specified by the user is extracted and the appropriate bit is set in INPUTMRG.

If any errors are detected, control is passed to CCERR in overlay 1. At the end of a card, control is returned to SCAN.

OTFSCN, AD-D2

MSGADD is updated with the address of the OUTFIL control card (from TBLADD) for use in case of error conditions. A branch is then made to INF1.

MODSCN, AD-C3

MSGADD is updated with the address of the MODS control card (from TBLADD) for use in case of error conditions. The routine continues at MOD1.

MOD1, AD-G3

Field definers are located and a branch is made to the respective routines:

- CKFLD1 -- phase 1 and 3 user routines names, length, and exits used.
- CKFLD4 -- phase 4 user routines names, length, and exits used.

CKFLD1, AD-D4

If phase 1 or phase 3 field definers are present, the address of the appropriate phase table, PHASE1 -- PHAS34, is initialized. For a sort run, PHAS34 is used for phase 3. For a merge, PHAS34 is used for phase 4.

CKFLD4, AD-C5

Field definer PH4 is used only on a merge run. Values are placed in phase table PHASE34.

Each field definer is checked for name, address and exit value by routines labeled CKNAME, MVADDR, MVEXIT. If any errors are detected, control is passed to CCERR in overlay 1. A branch is made to SCAN when the end of the card is reached.

OPTSCN, AE-B1

MSGADD is updated with the address of the OPTION card (from TBLADD) for use in case of error conditions. The option code is placed in MANDBK and a branch is made to CKDEFS. If there is no option card, the branch is to ENDSCN.

CKDEFS, AE-E1

The field definers are located and the corresponding bits are set in the phase table:

Constant Label	Bit	Option
RESTAT	RESTAR	RESTART
STORAG	(value)	STORAGE
VERIFY	VERIFY	VERIFY
PRINT2	PRINT	PRINT
CALCAR	CALCAREA	CALCAREA
ADDRROT	RAF	ADDRROUT
LABEL	INPUTMRG- INPUTMRG+4	LABEL
CKKEY	(value)	KEYLEN

Note: PRINT, CALCAREA, and ADDRROUT options have been processed by overlay 1. STORAGE and KEYLEN definers make use of count routines to count number of bytes. LABEL option assumes standard input and output labels if not specified. This option must be compatible with the type of input.

ENDSCN, AE-D2

If no errors are detected in this overlay, the END card causes the end card code (D-character) to be placed in MANDBK+7, overlay 5 is fetched, and the program continues at DSORT005.

If any errors are detected, the program may not be able to continue, depending on the type of error. In such cases, a branch is made to ABORT (Chart AA); otherwise, overlay 10 is fetched.

will be contained within the two half work areas. Either half of the work area will contain at least one but not more than six of the sections. For example, if the first half work area consists of four sections, then the second half work area could consist of only three.

OPEN WORK AREA AND PRE-EDIT - AF,AG

Overlay 5 builds a table of the work area extents for use in phases 1, 2, and 3, unless the CALCAREA option has been specified, in which case the building of the table is bypassed and the program continues directly to calculate the phase sizes.

XTENT cards are set up, giving the unit, class, number, and location of the areas that are allocated for the user work area. The information from the cards is extracted from the supervisor program, and stored in XTAREA when FILEW DTFPH is opened. The total available tracks for the work area, computed using the limits given in the XTENT cards, need not be a single continuous extent, but can be a maximum of six separate areas.

Three tracks are subtracted from the total available work area to be used for checkpoint records. The remaining tracks are equally divided into two work areas and each is designated as tracks required (TR). If the original number of available tracks was even, a total of four tracks is subtracted even though only three tracks are required for checkpointing.

The work area table, labeled TABLEB, is divided into up to seven sections, which

The number of tracks available (TA) in the first XTENT card is compared to the tracks required (TR) for half the work area. The result is one of two conditions:

- $TA \geq TR$ -- TA equal to or greater than TR
- $TA < TR$ -- TA less than TR.

($TA \geq TR$) if TA (tracks available) in the extent limits is equal to or greater than TR (tracks required), the lower limits of the XTENT card are placed into the lower limit area of section 1 of TABLEB. The upper limit is computed using TR. Tracks left over from this extent are used with the TR of the second half work area (Figure 11).

($TA < TR$) If TA (tracks available) in the first extent card is less than the tracks required, the lower and upper limits of the first card are placed into the first section of the work area table. The tracks available in the next extent card are obtained and added to the tracks available from the first card. The lower limit of the second card is placed into the lower limit area of the next section. If the combined TA is greater or equal to TR, the upper limit of the second section is now computed. These two sections make up the first-half work area. Any remaining tracks are used with the second-half work area (Figure 12).

	Cylinder--Head	L	←	Limits	→	U	Cylinder--Head	
XTENT Card--1	00--06						01--08	13
XTENT Card--2	05--00						06--05	16

Total Tracks = 29 - 3 Check Point Tracks = 26.
1/2 Work Area = 13 Tracks (TR)

		C	H	H	R	L	←	Limits	→	U	C	H	H	R
TABLEB	Section 1 (1/2 WK)	0	06	0							1	09	0*	
	Section 1 (1/2 WK)	5	00	0							6	03	0*	

Note: 6031 is CKPT Address

*The upper limit track address is not used by the section.

Figure 11. Work Area Table (2 Sections)

	Cylinder--Head		Cylinder--Head		
XTENT Card 1	00--06	L ←-- Limits →-- U	01--04		9
XTENT Card 2	03--00	L ←-- Limits →-- U	03--06		7
XTENT Card 3	06--03	L ←-- Limits →-- U	06--09		7
XTENT Card 4	08--00	L ←-- Limits →-- U	08--08		9

} = 32 Tracks

Total Tracks = 32 -4 Checkpoint Tracks = 28
 1/2 Work Area = 14 Tracks (TR)

Address of
Next Disk Area

		C	HH	R	L ←-- Limits →-- U	C	HH	R	1	
TABLEB	Section 1	} 1/2 WK	0	--06	0	L ←-- Limits →-- U	1	--05	0	9
	Section 2		3	--00	0	L ←-- Limits →-- U	3	--05	0	5
TABLEB	Section 1	} 1/2 WK	3	--05	0	L ←-- Limits →-- U	3	--07	0	2
	Section 2		6	--03	0	L ←-- Limits →-- U	7	--00	0	7
	Section 3		8	--00	0	L ←-- Limits →-- U	8	--05	0	5

} = 14 Tracks
} = 14 Tracks

Note 8051 is CKPT

Figure 12. Work Area Table (5 Sections)

Note: In Figure 12, the upper limit of the XTENT card is included as part of the area required by that extent. The upper limit shown for Section 1 of TABLEB, for example, indicates the start of the next disk area and is not part of the work area Section 1. It is given only to illustrate the calculation.

The second-half work area is built in the same manner as the first-half work area.

With the work area table completed, the LOGWKA table is set up. This table contains addresses of the pointers, or bytes, indicating the first section of each half-work area.

Four tracks are left over from the total available tracks for checkpoint records (Figure 12). The upper limit of the final section in the second-half work area is the starting point for the checkpoint records.

DSORT005, AF-B1

The base registers are initialized and a branch is made to CLEAR.

Note: After base registers are initialized, the area from IBV005 to H5 (overlay number) is reserved for IOCS as a work area for the open routines (256 bytes).

CLEAR, AF-C1

One of three courses of action is taken, depending on the type of run:

- CALCAREA option - the processing for building the table of work area extents is bypassed by a branch to FIRSTED (Chart AG).
- MERGE - a branch is made to MGINIT.
- SORT - general register 4 is initialized to read a maximum of six extents and a branch is made to OPEN.

MGINIT, AF-D2

General register 4 is initialized to read one XTENT during file opening and a branch is made to OPEN.

OPEN, AF-F1

The work area file (FILEW) is opened and IOCS returns control at READX.

READX, AF-G1

A test is made to determine whether it is a sort or a merge run. For a merge, a branch is made to MRGOPN; for a sort, a branch-and-link is made to SORTXT. In both cases, the routine then resumes at READX1.

MRGOPN, AF-G2

MRGSW is turned on (unconditional branch) to allow analysis of only one XTENT; a branch is then made to READX1.

SORTXT, AF-H1

Extents are sorted by sequence and stored in XTAREA; a branch is then made to READX1.

Note: XTENTS may be given out of sequence as long as the symbolic assignments are in sequence. This feature makes possible the optimum placement of the work area for a faster sort. FILEW XTENTS must follow these rules if more than one extent is given:

1. XTENT cards must be in sequence by symbolic unit assignment. For example:

```

XTENT1    003 -- SYS005
           002 -- SYS006
           004 -- SYS007
           001 -- SYS008

```

2. Symbolic units must be assigned to unique devices. For example, SYS005 cannot be the same unit as SYS008, etc.
3. XTENTS may be out of sequence by extent sequence number as in item 1 (003, 002, 004, 001). This allows for optimum placement of work area sections for a faster sort time. The assignment phase will sort the given XTENTS by sequence number and the work area will be placed as the user designates according to the sequence numbers. For example, in the illustration in item 1 the first XTENT used to build the work area will be symbolic SYS008, followed by SYS006, SYS005, and SYS007, in that order. The checkpoint will reside on the device assigned to SYS007.

READX1, AF-K1

Index registers are initialized.

READXT, AF-B3

The first 12 bytes of each XTENT are moved to the assignment phase extent work area, XTENT. When FILEW was opened and each extent originally fetched, a W was placed in byte 0 of XTENT. A test is now made for the W in each extent. If found, the extent is repositioned for easier handling by the assignment phase. A branch is then made to SI1-4.

When no W is found, all the extents have been processed; a branch is made to SAVEWK to determine if at least four tracks were given for the last XTENT. If not, a branch is made to ER3075 to execute the required error routine. If at least four tracks were given, the total number of tracks given is placed in TRACK, from which 3 is subtracted (for checkpoint tracks). The routine then continues at METH1I.

SI1-4, AF-C3

The lower and upper limit information is extracted and converted to tracks. The difference between the lower and upper limits determines how many tracks are available in the current extent; this number is placed in XTAREA position. The total number of tracks given by user (TRACKS) is updated. After all FILEW extents have been processed, a branch is made to MRGCKP for a merge or to SAVEWK for a sort.

Note: For a merge run, the switch at MRGSW is on (unconditional branch) and the program would have branched to MRGCKP after processing one FILEW extent.

METH1I, AF-F3

After initializing registers and constants with extent information, the size of half work area is calculated and saved in HALFTK. TA (current extent card) is then compared to TR (tracks required in half work area.) If TA ≥ TR, a branch is made to INT1; otherwise, the branch is to INIT2.

INIT1, AF-H3

The section counter (SECCTR) is reset to indicate the first section of a logical half work area, the lower limit is placed

in TABLEB, and the number of current tracks available that are to be used for the section is determined.

Note: If TA is greater than TR, TR is added to the lower limit to compute the upper limit of this logical half work area.

Excess TA tracks are saved for a section of the next logical half work area. If the last section was just computed, the first excess track is saved for the first checkpoint record. The upper limit is then placed in TABLEB.

If it is not the last half work area, a branch is made to INITA1.

If there are no excess tracks (TA = TR), a branch-and-link is made to GETXTN, to get the next extent card information for the start of second-half work area, before branching to INITA1.

INITA1, AF-K3

Registers and constants are initialized to build the next section of the first logical work area, or a section of the second half logical work area, with TA (tracks available) remaining from the current extent card. If the remaining TA is less than TR, a branch is made to INIT2; if it is equal to or greater than TR, the branch is to INIT1+4.

INIT2, AF-G4

If TA is less than TR, the lower and upper limits are placed in the section of the current logical half work area. The next extent card information is obtained and the TA from this extent is added to the TA of the last extent. If the current TA is equal to or greater than TR, the upper limit is calculated and placed in the section of the current logical half work area. If the current TA is less than TR, a branch is made to INIT2 for the next section of this logical halfwork area. If the current TA is greater than TR, a branch is made to INIT1+4.

EXIT, AF-B5

When it is determined that the work area is completely built, switches END1, END2, and END3 are initialized to exit from INIT1 or

INIT2. The program then branches to EXIT, where the displacement pointers of TABLEB are scanned to determine the beginning of each logical half work area. The displacement values are reset to their original values and placed in LOGWKA. Zeros, used to close pointers, are inserted in the displacement pointer immediately following the last displacement pointer in the last section of the second-half work area table initialized (TABLEB). A branch is then made to SAVCKP. This signals phases 2 and 3 that the end of one half of the work area has been reached and forces processing of the other half. For example:

0000 - first half pointer
000C - second half pointer
0000 - end of table

Phase 2 fills the first half (0000), the next pass fills the second half (000C). The next pass detects 0000, which points back to the first half. The process is repeated until all passes are complete.

SAVCKP, AF-E5

The checkpoint disk track is computed using the upper limit of the last section used in the work area, and is saved in DOUBLE+26 for SYMBOLUNIT displacement and in DOUBLE+28 for CHHR. A branch is then made to FIRSTED (Chart AG).

MRGCKP, AF-D4

A test is made to determine if at least two tracks were given for checkpoint for a merge. If not, an error condition results. If two or more tracks are given, the merge checkpoint disk address is calculated and a branch is made to FIRSTED (Chart AG).

OPEN WORK AREA AND PRE-EDIT (CONTINUED) - AG

Constants from sort control cards, to be used in the calculation of phase sizes, are edited. After checking that these constants are included, the information needed for phase size calculation is retrieved and processed. Some of the information is:

- Format (Floating point, unsigned binary, fixed point, packed decimal, or zoned decimal).

- Fields (control data specification).
- Length (record).
- Block size (input and output).
- Block type (fixed or variable).
- Record type (fixed-length or variable-length).
- Type of run (sort or merge).
- Type of input and output (disk or tape or mixed).
- OM for merge and FILES for sort.

Each phase size is computed as determined by the factors in the control cards, such as the record type (variable-length, fixed-length, ADDRROUT), format, number of control fields, etc. After all phase sizes are calculated, they are stored in their respective constants: PHS1, PHS2, and PHS34 (phase 3 for sort or phase 4 for merge). Maximum sizes for phases 2 and 3 are calculated and stored in PH2MX and PH34MX, respectively. The length of the user area is computed, and the result is saved in LEN1 for phase 1 and in LEN34 for phases 3 or 4.

Phase 4 is computed for a merge-only run.

Formulas not explained in the following text are in IBM System/360 Disk Operating System Sort/Merge Program Specifications, Form C24-3444, unless otherwise stated.

FIRSTED, AG-B1

Information used in phase size calculations is edited and tested to ensure that all mandatory information has been given. Any mandatory information missing results in an error condition. Items checked are:

- Validity of L5 for sort of variable-length records (FIRSTED) - if records are variable-length and ADDRROUT is not specified, a test is made to determine if average record length was given and is valid. L5 must be equal to or less than L1.
- Format code (FMAT) - check if format was given in sort or merge control statement.
- Record length (CLANDB) - check if length was given.
- Input block size (CKBK) - check if input block size was given.

- Output block size (CKBKO) - check if output block size was given.
- Number of control fields (CF) - check if FIELDS was given.
- Block type (RT1) - if FILES=1, check to determine if block type was specified. If FILES>1, assume variable blocking internally for sort. If block type was not specified, variable blocking is assumed.
- Input and output device (CKOMED) - determine if input and output device type was specified.
- L4, minimum number of bytes in a single logical input record for variable-length records (RCT) - if record type is variable and ADDRROUT was not specified, check to determine if L4 is less than L5.
- Maximum internal OM for sort (BLK) - check for specification of fixed blocking of variable-length records.
- Validity of FILES (OMVAL) - check for validity of number of files for sort (maximum of 9) or for merge (maximum of 4).

If no errors are detected in the foregoing checks, the routine continues at CORVAL. If any errors are detected, the program branches to the appropriate error routine. (See program listing for specific error routine used.)

CORVAL, AG-E1

CORESZ is checked for zeros. If zero, no storage entry was specified and the actual machine size, retrieved from the communications region, is used. In either case, tests are made to determine that the size is between 16,384 and 524,288 bytes. If less than 16,384, an error condition results. If more than 524,288, an information message is printed. Only a sort run utilizes 524,288 bytes.

Starting at the label BLK, the maximum internal order of merge is calculated for a sort run:

- Variable-length records - 16K = 6 way OM
- Fixed-length records or ADDRROUT - 16K = 7 way OM

The routine then continues at P1SIZE.

For a merge run, the OM initialization is bypassed by a branch to BLK2.

P1SIZE, AG-G1

Registers are loaded with phase 1 table address and basic phase 1 size.

Note: Figure 13 contains the basic sizes for all phases, as well as the formulas for calculating sizes for the several types of runs with either fixed- or variable-length records.

BLK2, AG-F2

Phase 4 size (PHS4) is placed in PHS34, phase 4 switch (P4NAMS) is turned on (unconditional branch), and a branch is made to P4NAM.

P4NAM, AG-H1

Values N and M are calculated for phases 1, 2, and 3:

- $N = \text{number of control fields times ten.}$
- $M = 6(L1/256)$ for phases 1 and 2;
 $6(L3/256)$ for phase 3.

For a merge-only run, a branch is now made to PHSIZ; for a sort run, the routine continues at STDFV.

P4SIZ, AG-J2

Phase 4 size (PHS34) is adjusted with M and N. The conversion routine lengths are reinitialized for a merge-only run, and a branch is made to USESTR.

STDFV, AG-K1

The calculations for phase 1 size are completed using separate routines for:

- Variable-length records with ADDRROUT
- Variable-length records without ADDRROUT
- Fixed-length records with ADDRROUT
- Fixed-length records without ADDRROUT

P2SIZE, AG-A3

Phase 2 size (see Figure 13) is calculated for the type of record used.

P3SIZE, AG-B3

Phase 3 size (see Figure 13) is calculated for type of record used and the routine branches to USESTR.

USESTR, AG-C3

The length of the user area in phase 1 (LEN1) and in phases 3 and 4 (LEN34) are calculated, if not given by the user.

- $LEN1 = \text{CORESZ} - \text{ADD1}$ (Rounded to next higher double word boundary.)
- $LEN34 = \text{CORESZ} - \text{ADD34}$ (Rounded to next higher double word boundary.)

The user address is checked to be sure it is less than CORESZ.

CF256, AG-D3

The total length of all control fields is calculated, checked for validity, and saved in TLACFD.

Phase 1 BASIC = 2744 Phase 2 BASIC = 1526
Phase 3 BASIC = 1426 Phase 4 BASIC = 2238

N = No. of control fields

M = (L/256) L = L1 (phases 1 and 2), L3 (phase 3)

CFADD = 84 when N > 1 (SORT) CFADDM = 66 when N > 1 (MERGE)

PHASE 1	Fixed W/O ADDRUT = BASIC + 10N + 6M + DC + Q				
	Fixed W/ ADDRUT = BASIC - TAGDIF + 10N + 6M + DC + Q ₁				
	Variable W/O ADDRUT = BASIC + 10N + DC + Q				
	Variable W/ ADDRUT = BASIC - TAGDIF + 10N + DC + Q ₁				
	DC =	FI 76	FL 120	ZD 136	PD 180
TAGDIF =	88				
Q =	208	If no addrut and if any of the following conditions occur, add Q once: Other than BI format; variable records; user routines PH1.			
Q ₁ =	128	If any of the following conditions occur, add Q ₁ once: ADDRUT option; other than BI format; user routines PH1.			
PHASE 2	Fixed with or without ADDRUT = BASIC + 10N + 6M + CFADD (OM 2-4 DSORT201)				
	or Variable with ADDRUT = BASIC + 902 + 10N + 6M + CFADD (OM 5-7 DSORT202)				
	Variable without ADDRUT = BASIC + 186 + 10N + CFADD (OM 2-3 DSORT203)				
	" = BASIC + 186 + 928 + 10N + CFADD (OM 4-6 DSORT204)				
PHASE 3	Fixed with or without ADDRUT = BASIC + 10N + 6M + DC ₃ + CFADD (OM 2-4 DSORT301)				
	or Variable with ADDRUT = BASIC + 834 + 10N + 6M + DC ₃ + CFADD (OM 5-7 DSORT302)				
	Variable without ADDRUT = BASIC + 134 + 10N + DC ₃ + CFADD (OM 2-3 DSORT303)				
	" = BASIC + 134 + 848 + 10N + DC ₃ + CFADD (OM 4-6 DSORT304)				
DC3	FI = 60	FL 94	ZD 114	PD 154	
PHASE 4	All cases = BASIC + 10N - DC ₄ + CFADDM				
	DC4	FI 50	FL 102	ZD 164	PD 274

Figure 13. Disk Sort/Merge Phase Size Formulas

For other than an ADDRROUT run, a branch is made to TSTER5. For an ADDRROUT run, the validity of L3 (maximum length of a single output record) is checked, and the tag type and Exit 32 are initialized. The routine then branches to TSTER5.

TSTER5, AG-G3

If any errors are detected in this overlay, a branch is made to GETEOJ to fetch overlay 10. If no errors occurred, overlay 6 is fetched.

COMPUTE MAXIMUM ALLOWABLE INPUT AND OUTPUT RECORD AND BLOCK LENGTHS -- AH

Lmax and Bmax are calculated for each phase, using the phase sizes just calculated plus core size, supervisor, and, if applicable, the user area length. Because Lmax and Bmax must fit into all phases, the smallest calculated value in the respective calculations is taken by the program as LMAX, BMAX.

BMAX must be considered when minimum-length records are used, and the 8-byte address for each record becomes a determining factor for the number of records that can be read into phase 1. An increased number of minimum-length records results in using more core space for the address table, thereby reducing the space available for record processing.

LMAX, BMAX is calculated for phase 4 only for merge-only runs.

NOTE: Refer to the Systems Reference Library publication, IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444, for additional descriptions of the formulas used throughout this overlay.

DSORT006, AH-B1

The registers to be used in overlay 6 are initialized and the routine continues to CLORBV.

For a merge-only run, a branch is made to BLMAX4 to compute LMAX and BMAX for fixed- or variable-length records.

For a sort run, a branch is made either to VL1 for variable-length records without ADDRROUT, or to L1 for fixed-length records or ADDRROUT run.

BLMAX4, AH-C2

The maximum record length (LMAX) and maximum block length (GMAX) are calculated for fixed- or variable-length records.

$$LMAX = BMAX = CORESZ - [SUPER + PHS34 + LEN34 + BLKSOZ + (OM \times BLKSIZ) + L13]$$

- CORESZ Core size.
- SUPER Length of supervisor.
- PHS34 Length of phase 4 program.
- LEN34 Length of phase 4 user area.
- BLKSOZ Output block length.
- OM Order of merge (FILES).
- BLKSIZ Input block length.
- L13 (L1-L3), when L1 greater than L3.

A branch is then made to CKLMAX.

L1, AH-E2

Register 14 is initialized to branch to CFL3MAX. LMAX for fixed-length records is calculated for phases 1, 2, and 3 and stored in STP123, STP123+4, and STP123+8, respectively.

- $PH1 \text{ LMAX} = \frac{CORESZ - (SUPER + PHS1 + LEN1 + 8) *}{2}$
- SUPER Sort program origin.
- PHS1 Computed length of phase 1 program.
- LEN1 Computed length of phase 1 user area.
- 8 Two 4-byte address table entries.
- 2 Allows at least two areas (input and output).

- $PH2 \text{ LMAX} = \frac{CORESZ - SUPER - PHS2}{3} *$

* Rounded to the next lower whole number.

PHS2 Length of phase 2 program.
 3 Minimum OM (2) + 1 for output.

•
$$PH3\ LMAX = \frac{CORESZ - (SUPER + PHS34 + LEN34)}{3} *$$

PHS34 Length of phase 3 program.
 LEN34 Length of phase 3 user area.
 3 Minimum OM (2) + 1 for output.

A branch is then made to DETSMALL.

VL1, AH-F1

Register 14 is initialized to branch to CVL3MAX. LMAX for variable-length records is calculated for phases 1, 2, and 3 and stored in STP123, STP123+4, and STP123+8, respectively.

•
$$PH1\ LMAX = \frac{CORESZ - (SUPER + PHS1 + LEN1 + 8)}{3}$$

CORESZ Core size.
 SUPER Sort program origin.
 PHS1 Computed length of phase 1 program.
 LEN1 Computed length of phase 1 user area.
 8 Two 4-byte address table entries.
 3 Allows for 1 input, 1 output, and 1 overflow area.

•
$$PH2\ LMAX = \frac{CORESZ - SUPER - PHS2}{5}$$

PHS2 Computed length of phase 2 program.
 5 Minimum OM (2) + 1 each for overflow, input, and output.

•
$$PH3\ LMAX = \frac{CORESZ - (SUPER + PHS34 + LEN34)}{5}$$

PHS34 Computed length of phase 3 program.
 LEN34 Computed length of phase 3 user area.

A branch is then made to DETSMALL.

DETSMALL, AH-G2

The LMAX values calculated for phases 1, 2, and 3 (either fixed-length or variable-length, as the case may be) are compared to each other to determine the smallest value. This value is stored in

LMAX and is the maximum input record length that will fit in all phases. The routine then continues at CKLMAX.

CKLMAX, AH-H2

If the computed LMAX is greater than 3624 for ADDRROUT or fixed-length records, or greater than 3620 for variable-length records, LMAX is reduced to 3624 or 3620, as the case may be. A branch is then made to the address contained in register 14 (CFL3MAX for fixed/ADDRROUT or CVL3MAX for variable).

CFL3MAX, AH-J1

Register 10 is initialized with the address of CKBI. L3MAX for fixed-length records is calculated; this is the maximum record length that phase 3 can process. This calculated value is compared to that given by the user to determine if the user-given value is within the limits compatible with storage size and the sort program.

$$ROL = CORESZ - SUPER - PHS34 - LEN34 - 2(LMAX) \leq 3624$$

A branch is then made to CKL3MAX.

CVL3MAX, AH-K2

Register 10 is initialized with the address of VB1. L3MAX for variable-length records is calculated and compared as described in CFL3MAX. The only difference is in the formula:

$$ROL = CORESZ - SUPER - PHS34 - LEN34 - 4(LMAX) \leq 3620$$

A branch is then made to CKL3MAX.

CKL3MAX, AH-K1

The calculated maximum phase 3 record length is compared to 3624 and the smaller value is used for L3MAX. A branch is then made to the address stored in register 10.

CKBI, AH-A4

A branch is made to one of three locations, depending on the type of run:

- B1 - for fixed-length records with fixed blocking, or
- VARBI - for fixed-length records with variable blocking, or
- RAFBI - for ADDRROUT sort

VARBI, AH-A5

Phase 1 block length for fixed-length records with variable blocking is calculated and stored in BLK123, and a branch is made to B2.

$$PH1 \text{ BMAX} = \frac{CORESZ - (SUPER + PHS1 + LEN1 - L1)}{3 + (8/L1)} *$$

L1 Maximum number of bytes in a single logical input record

RAFBI, AH-C5

Phase 1 block length for fixed-length records with ADDRROUT output is calculated and stored in BLK123, and a branch is made to B2.

$$PH1 \text{ BMAX} = \frac{CORESZ - (SUPER + PHS1 + LEN1)}{3 + (8/CF+10)}$$

CF Total length of all control fields.

B1, AH-C4

Phase 1 block length for fixed-length records in fixed blocks is calculated and stored in BLK123, and the routine continues to B2.

$$PH1 \text{ BMAX} = \frac{CORESZ - (SUPER + PHS1 + LEN1)}{2 + (8/L1)} *$$

L1 Maximum number of bytes in a single logical input record.

B2, AH-D4

The block length for fixed-length records of all types is calculated for phases 2 and 3, and stored in BLK123+4 and BLK123+8, respectively.

$$PH2 \text{ BMAX} = \frac{CORESZ - SUPER - PHS2}{3} *$$

$$PH3 \text{ BMAX} = \frac{CORESZ - (SUPER + PHS34 + LEN34)}{3} *$$

Register 14 is initialized with the address of CFB3MAX and a branch is made to DETLOWER.

VB1, AH-B3

The block lengths for phases 1, 2, and 3 are calculated and stored in BLK123, BLK123+4, and BLK123+8, respectively.

$$PH1 \text{ BMAX} = \frac{CORESZ - (SUPER + PHS1 + LEN1)}{3+(8/L4)} *$$

$$PH2 \text{ BMAX} = \frac{CORESZ - [SUPER + PHS2 + (2 \times L1)]}{3} *$$

L1 Maximum number of bytes in a single logical input record.

$$PH3 \text{ BMAX} = \frac{CORESZ - [SUPER+PHS34+(2 \times L1)+LEN34]}{3} *$$

Register 14 is initialized with the address of CVB3MAX and a branch is made to DETLOWER.

DETLOWER, AH-D4

The calculated block lengths for phases 1, 2, and 3 (either variable-length or fixed-length, as the case may be) are compared to each other to determine the lowest value. This value is stored in BMAX and is the maximum allowable block length for sort. The routine then continues at CKBMAX.

* Rounded to the next lower whole number.

CKBMAX, AH-D3

The BMAX determined, as described under DETLOWER, is compared to 3624 and the smaller value is stored in BMAX. A branch is then made to the address stored in register 14 (CFB3MAX for fixed/ADDRROUT or CVB3MAX for variable).

CFB3MAX, AH-F4

B3MAX for fixed-length records is calculated; this is the maximum block length that phase 3 can process. This calculated value is compared to that given by the user to determine if the user-given value is within the limits compatible with storage size and the sort program.

BOL =
CORESZ-SUPER-PHS34-LEN34-L13-2(BMAX) ≤ 3624

The routine then continues to FETCH7.

CVB3MAX, AH-F3

B3MAX for variable-length records is calculated and compared as described in CFB3MAX. The only difference is in the formula:

BOL =
CORESZ-SUPER-PH34-LEN34-2(L1)-2(BMAX) ≤ 3624

The routine then continues to FETCH7.

FETCH7, AH-G4

If no errors are detected in this overlay, overlay 7 is fetched. If errors are detected, a branch is made to FETCHEND to fetch overlay 10.

POST EDIT - AJ

Overlay 7 completes the editing of the sort control cards, and of the computed results of the control card data. Control-field lengths cannot exceed a total of 256 bytes. Multiple control fields are checked for overlapping on all formats except unsigned binary. Computed maximum record-lengths are checked against user-given record-lengths. If user-given L3 is greater than L1, a test is made to see that

exit 32 (for sort) or exit 43 (for merge) is specified. (These exits can be used to shorten or lengthen record lengths.) When non-standard labels are specified, exit 31 (for sort) or 44 (for merge-only) must be specified.

Other checks are made for:

- Given mandatory file size and validity of same.
- Current tape assignments when tape input or output is specified for sort or merge.
- Input/output media and labels specified.
- Validity of NOTPMK option.
- Valid sort or merge volume entries.
- Validity of key length entries.
- Tape options.

DSORT007, AJ-B1

Registers are initialized for overlay 7 and the routine continues at NORAF.

NORAF, AJ-C1

For a sort of variable-length records, the routine is initialized to use L4 (minimum length of a single logical input record) instead of L1 for checking. A branch is then made to GETL1+4.

For all other types of runs (ADDRROUT sort, fixed-length record sort, or merge-only), a branch is made to GETL1 to initialize the routine to use L1 (maximum length of a single logical input record) for checking.

GETL1, AJ-D1

A register is loaded with the address of the phase table area, CFLLNG, which contains information pertaining to the control fields. Several validity checks are then made:

- The length of any control field cannot exceed the end of a record.
- The byte total of all control fields cannot exceed 256.

- Floating-point format fields cannot be greater than 8 bytes, and zoned or packed decimal cannot be greater than 16 bytes, including the sign.

The control field length and location is incremented by 1 to obtain the original value (the value prior to overlay 3) but the table values are not changed.

L3L1, AJ-E1

Note: For an ADDROUT run, the function provided at this location would have been executed in overlay 5 and a branch is now made to LENHI.

If L3 (output length) is less than L1 (input length), it indicates that the user desires to shorten records; exits 32 or 43 must be specified for this purpose or an error results for variable-length records. Tests are made for the type of record and for sort or merge-only run. The program automatically truncates the low-order bytes of fixed-length records if L3 < L1 is specified. For variable-length records, the truncating must be done by the user by means of exit 32 for a sort or exit 43 for a merge. The routine then continues at LENHI.

LENHI, AJ-G1

L1 (maximum number of bytes in a single logical input record), located in INTPRL, is tested to determine that it is equal to or less than the calculated LMAX. If L1 is greater, an error message is printed. The routine then continues to CKL3MX.

CKL3MX, AJ-H1

For a merge-only run, a branch is made to LENLO.

For a sort, the switch at LENLOSW is turned off (no-op). L3 (maximum number of bytes in a single logical output record), located in INTPRL+4, is compared to calculated L3MAX. If L3 is greater than L3MAX, a branch is made to LENLOA+4 to print the error message. If L3 is equal to or less than L3MAX, a branch is made to LENLO.

LENLO, AJ-J1

For a merge, the number of inputs allowed is obtained from location OM and loaded in register 4. For a sort, register 4 is loaded with one. For a merge or a sort, register 3 is loaded with the contents of INPUTMRG.

The routine then continues at LOOP.

LOOP, AJ-A3

INPBK is initialized with the type of input specified. For a sort, only one input type (tape or disk) will be indicated in this byte. For a merge, INPBK can indicate mixed input types. INPBK is used later when checking options relating to device types.

TESTDEV, AJ-B3

This routine determines if system assignments for tape devices are compatible with input/output devices specified for the sort/merge program.

A file count is set up in general register 4; this count is equal to the number of input files (FILES), plus 1 for the output file. Devices are extracted from the PUB table for SYS001 through SYS010, depending on the number in FILES. When the count in general register 4 is reduced to zero, indicating that all devices have been extracted, the remainder of this routine is bypassed and the program branches to VALIDATE. If tape was not specified for any input or output, the program branches to MINBLKI.

For merge-only: The registers used are initialized:

- R2 - SYSTABLE pointer
- R4 - file count
- R5 - device pointer to INPUTMRG
- R6 - LUB pointer

The low-order bit of INPUTMRG is tested to determine if the device is tape or disk. If the device is tape, register 9 is zeroed (at GETDEV) and register 8 is loaded with the LUB address. Register 8 is then incremented by 22 (SYS001 is the eleventh 2-byte entry in the LUB table) plus the value contained in register 6 (zero for first time through) to obtain the address of the LUB entry. The LUB entry (PUB pointer) is then inserted into register 9

and multiplied by 8 (number of bytes displacements per PUB) to obtain the address of the first PUB on the channel list (PUB address). The channel and unit identification, and the device type for SYS001 are then extracted from the PUB address. For a more detailed description, refer to the Program Logic Manual publication, IBM System/360 Tape Operating System, System Control, Form Z24-5022.

Register 2 contents are increased by 4 to obtain the contents of SYSTABLE+4 (00000002). The contents of register 6 (zero) are increased by 2 (number of bytes per LUB) to obtain the address of the next LUB entry. The contents of register 5 (device pointer) are increased by one to indicate the first input device, FILEA, (INPUTMRG+1). The file count, contained in register 4, is then decreased by one.

The program branches to LOOP1 to test the low order bit of INPUTMRG+1 for disk or tape for SYS002. The processing described in the preceding paragraph is repeated until the file count in register 4 is reduced to zero, at which time a branch will be made to VALIDATE.

If the device checked (in register 5) is not tape, the routine to obtain channel, unit, and device type (at GETDEV) is not entered but registers 2, 4, 5, and 6 are altered, for the next SYSTABLE entry, as described for tape device identification.

For sort: The compatibility check for a sort run is similar to that described for a merge. The major difference is that the switches at SS1 and SS2 are set on (unconditional branch) so that the contents of register 5 (output file data from location INPUTMRG) are altered only once for input file (FILEA) data from location INPUTMRG+1. Specifications for a sort state that all input files (maximum of 9) must be of the same type. It is unnecessary, therefore, to continue to alter register 5 after processing the first input file, because compatibility checks are not required for SYS002 through SYS010. General registers 2, 4, and 6 are altered for each SYSTABLE entry and processing is repeated as described for a merge. A branch is then made to VALIDATE.

VALIDATE, AJ-B4

Registers 4, 5, and 6 are loaded with zeroes and if the CALCAREA option is specified, a branch is made to MINBLKI. For all other type runs, register 2 is initialized for SYS001 and register 5 for output file (INPUTMRG). The output file is

tested for tape device. If the output file is not tape, registers 2 and 5 are altered for SYS002 and for the first input file, FILEA (INPUTMRG+1), respectively. INPBK is tested to determine if any input file is tape and, if not, the program branches to ISMERG.

If the output file is tape, one is added to the count in register 4, and INPBK is tested to determine if any input file is tape. If not, the program branches to ISMERG.

If any input file is tape, the count in register 4 is updated with the number of input files (FILES). If any device that should be a tape device is not assigned as such, an error message is printed to indicate the symbolic assignment in error.

ISMERG: For a merge, the first input file (FILEA) is checked to determine the device type. If tape, SYS002 is checked for tape assignment and for duplicate assignment (error). Registers 2 and 5 are altered for SYS003 and for the second input file, FILEB (INPUTMRG+2), respectively, and the checking routine is repeated for all input files. If any input file is disk and not tape, the check for tape assignment and duplicate assignment is bypassed for that particular file, but registers 2 and 5 are altered for the next file. When all files have been checked, a branch is made to MINBLKI. If any duplicate assignments are detected, an error message is printed.

For a sort, the switches at locations SS4A, SS2A, and SS5A are set on and the switch at location MS1 is set off. This modifies the processing so that the input file data contained in register 5 is not altered for checking SYS003 through SYS010. When all input files have been checked, a branch is made to MINBLKI.

MINBLKI, AJ-C3

The input and output block sizes for tape are checked to determine that they are equal to or greater than the minimum specified size of 12 bytes. If so, a branch is made to BLKHIP. If errors are detected, a branch-and-link is made to ERR79 to print an error message, after which the program returns to BLKHIP.

BLKHIP, AJ-D3

The calculated maximum block size is checked to determine if it is at least the minimum of 300 bytes. The routine then continues at BLKOK.

BLKOK, AJ-E3

For a merge, this routine is bypassed because its function has already been performed by overlay 6; a branch is made to CKSIZ.

For a sort, input and output block sizes are compared to computed maximums to determine their validity. For variable-length records, input and output block sizes are each checked to ensure that a 4-byte block size indicator is included. The routine then continues at CKSIZ.

CKSIZ, AJ-H3

A test is made to determine if file size was given for sort. The computed file size is then checked against the given size.

If the ADDRROUT option is specified for a merge run, an error message is printed.

The routine then continues at L1MULT.

L1MULT, AJ-J2

Tests are made to determine if L1 is a multiple of the given input block size and if L3 is a multiple of the given output block size. These tests are made only for fixed length records with or without ADDRROUT option.

Note: For "A" type ADDRROUT (address only), L3 must be equal to 10. For "D" type ADDRROUT (address plus control field), L3 can be less than the total control field lengths plus 10 (TLACFD+10), but not less than 11; L3 cannot be greater than TLACFD+10 unless exit 32 is specified.

OVLPCF, AJ-K2

Tests are made to determine if control fields overlap, only if there is more than one control field and if the format is other than unsigned-binary. The first control field is checked to see if it overlaps into any one of the following control fields. After checking the first field, the second is checked against the rest and so on, until all fields are compared. The routine then continues at C. If an overlapping field is found, an error message is printed before continuing to C.

C, AJ-A5

A check is made for valid placement of the control fields for variable-length records only. A control field may not start prior to byte 5 (bytes 1 through 4 are reserved for the record length indicator).

CKLABELS, AJ-B5

This routine checks:

- if labels were specified (if not, assume standard labels for all files)
- if assumed or given labels are valid for disk input and output.
- if input and output label types are given and that the types are compatible.

A merge may have mixed input (tape and disk) and, therefore, mixed labels. A check is made to determine that standard labels only are given for any disk input files.

The NOTPMK option may be specified at any time, for tape output. If the NOTPMK option is specified with standard labels, however, a warning message is printed. The processing will not be interrupted but a tape mark is written in later phases.

CKRWD, AJ-C5

The OPEN or CLOSE options are checked for tape input and output. If either or both options are not specified, RWD (rewind) is assumed.

CKTYPEIN, AJ-D5

For a merge, a branch is made to CKMERGE. For a sort with disk input, tests are made for any tape options that may have been given in error. Similar tests are made for output.

CKMERGE, AJ-E4

Merge-only options are tested for mixed input and for any tape options that may have been given in error for disk input.

LABEXCK, AJ-F5

Specified labels are checked against any user exits that may be required (see Figure 14). These checks are bypassed by branching to CKKEY if:

- All types of labels specified.
- Unlabeled files not specified.
- Both unlabeled and standard labels specified.

For a merge of unlabeled files, error conditions result if:

- Exit 41 specified for input.
- Exit 44 specified for output.

For a sort, error conditions result if:

- Exit 11 specified for unlabeled input files.
- Exit 13 specified for disk input files.
- Exit 31 not specified for non-standard output labels.

- Exit 31 specified for unlabeled output file.

The routine continues at CKKEY.

CKKEY, AJ-G5

The KELEN option may be used only for fixed-length records that are unblocked and, in the case of input, for disk only. Output may be on either tape or disk, and ADDROUT output may be blocked or unblocked. If these requirements are not met, an error message is printed. The routine then continues to CKVOL.

CKVOL, AJ-H5

Each input file is checked for the number of volumes (obtained from MRGVOL). The number of volumes is then set depending on several conditions:

- Standard labels - Use number given by user; if not given, use 0.
- Non-standard labels with exit 11 or 41 - Use 0 whether or not number is given.
- Non-standard labels without exit - Use number given by user; if not given, use 1.
- Unlabeled - Use number given by user; if not given, use 1.

When all files have been checked, a branch is made to TSER7.

TSER7, AJ-J5

If no errors were detected in this overlay, overlay 8 is fetched. If errors were detected a branch is made to FETCHEND to fetch overlay 10.

	EXIT #	USED FOR	REQUIRED	OPTIONAL	NOT ALLOWED
SORT	11	Process User or Non-std Labels-Phase 1	Never	With Std, User or Nonstd Labels	With Unlabeled Files
	12	All Purpose Phase 1	Never	Anytime	Never
	13	Bypass Unreadable Tape Input Blocks Phase 1	Never	With Tape Input Only	With Disk Input Only
	31	Process User or Non-std Labels Phase 3	With Non-std Labels	With Std, User or Nonstd Labels	With Unlabeled Files
	32	Record Altering Phase 3: Fixed Records, D-type Addrout, (Forv) Variable Records	When $L3 > < 1$ When $L3 > (CF+10)$ When $L3 \neq L1$	Anytime	Never
MERGE	41	Process User or Non-std Labels Phase 4 (Input)	Never	With Std, User or Nonstd Labels	With Unlabeled Files
	42	All Purpose Phase 4	Never	Anytime	Never
	43	Record Altering Phase 4: Fixed Records, Variable Records	When $L3 > L1$ When $L3 \neq L1$	Anytime	Never
	44	Process User or Nonstd (Output) Labels Phase 4	With Non-std Labels	Std, User or Nonstd Labels	With Unlabeled Files
	45	Bypass Unreadable Tape or Disk Input Blocks	Never	Anytime	Never

Figure 14. Exit Chart

COMPUTE CONSTANTS FOR FIXED-LENGTH RECORDS
- AK, AL

This routine (part of overlay 8) calculates constants for fixed-length records for a sort run. It also calculates some disk

input or output constants for a merge-only run.

Formulas used in the calculations are described in detail in IBM System/360 Disk Operating System, Sort/Merge Specifications, Form C24-3444.

DSORT008, AK-B1

saved but the value of the denominator is.

The base register is initialized and a branch is made to CLEAR.

CLEAR, AK-C1

Registers are cleared and a branch is made to the routine for the type of record to be processed:

- RAFRTN (Chart AN), for ADDRROUT run.
- VARRTN (Chart AM), for variable-length records.
- STEP (Chart AK), for fixed-length records.

If CALCAREA option is specified, switch CALCSW2 is turned on (unconditional branch).

STEP, AK-G1

For merge-only, a branch is made to OUTPT (Chart AL). For fixed blocking, a branch is made to STEP1. For variable-blocking, the switch at C2LP8SW is set to prevent storing the result of the first calculated value for PH1B1. In this case, the denominator is calculated and stored and the routine continues at CALPH1B2. (Also see note at end of STEP1.)

STEP1, AK-J1

The maximum number of records in a phase 1 sort block (PH1B1) is calculated:

$$PH1B1 = \frac{CORESZ - PHS1 - SUPER - LEN1}{2(L1) + 8}$$

- CORESZ Number of bytes of main storage available for sort program.
- PHS1 Calculated phase 1 program size.
- SUPER Size of supervisor.
- LEN1 Number of bytes required for user program in phase 1.
- L1 Number of bytes in a single or logical input record.

Note: The value of the denominator, 2 (L1) + 8, is stored in CP2LP8 for use in later calculations. For variable-blocking, the result of this calculation of PH1B1 is not

CALPH1B2, AK-K1

The alternate PH1B1 (PH1B2) is calculated and compared to that calculated in STEP1. The smaller value is saved as the effective PH1B1.

$$PH1B1 = \frac{3624}{L1}$$

Note: The result of this calculation is temporarily stored in PH1B2.

PH1B2 is then calculated:

$$PH1B2 = \frac{CORESZ - (PHS1 + BLKSIZ - L1 + SUPER + LEN1)}{CP2LP8}$$

BLKSIZ Input block size in bytes.

The result of this calculation is compared with that of the alternate PH1B2 (same as alternate PH1B1) and the lesser value is saved as the effective PH1B2.

STEP1B, AK-A4

The maximum number of bytes in a phase 2 block and in a phase 3 block are calculated and the smaller of the two values is saved in MAXBL for later calculations.

If the calculated MAXBL exceeds the maximum number of bytes per 2311 track (3624), the calculation is invalid and 3624 is substituted in MAXBL. The MAXBL of 3624 is rounded to a multiple of the input record length (L1) and retained in SBSIZE (computed sort block size).

For phase 3, if the input record length is equal to or less than the output record length (L1 ≤ L3), then:

$$MAXBL = \frac{CORESZ - (8 + PHS34 + BLKSOZ + SUPER + LEN34)}{OM}$$

For phase 3, if the input record length is greater than the output record length (L1 > L3), then:

$$MAXBL = \frac{CORESZ - (8 + PHS34 + BLKSOZ + L1 - L3 + SUPER + LEN34)}{OM}$$

- BLKSOZ Output block size in bytes.
- PHS34 Length of phase 3 or 4 program.

LEN34 Length of user program in phase 3 or 4.

COMBPT, AK-F4

For phase 2:

$$\text{MAXBL} = \frac{\text{CORESZ} - (\text{PHS2} + \text{SUPER})}{\text{OM} + 1}$$

Constants are calculated for disk operation:

- BYTBLK (bytes per block including disk gaps).
$$\text{BYTBLK} = 61 + [(537(\text{SBSIZE} + 1)) / 512] *$$
- BYBKLS (bytes per last block for disk).
$$\text{BYBKLS} = 1 + \text{SBSIZE}$$
- BPT (maximum number of blocks per 2311 track).
$$\text{BPT} = 1 + [(3625 - \text{BYBKLS}) / \text{BYTBLK}] *$$

STEP3, AK-B4

The maximum number of records that a phase 2 or phase 3 block can contain (RECBLK) is calculated:

$$\text{RECBLK} = \frac{\text{SBSIZE}}{\text{L1}}$$

CKBPT, AK-G4

The calculated BPT is compared to the previous BPT. If the calculated value is greater, a branch is made to CBYPTK; if not, RECBLK is checked to see if it equals one. If RECBLK equals one, and the CALCAREA option is not specified, a branch is made to SW1; with the CALCAREA option, the branch is to USEORIG. If the calculated BPT is less than the previous BPT but greater than 1, the routine continues at REDUCE.

STEP3A, AK-C4

If calculated RECBLK is equal to or less than PH1B2, a branch is made to STEP4. If RECBLK is greater than PH1B2, RECBLK must be recalculated.

Calculate the largest multiple of number of records per input block (BI) that is equal to or less than PH1B1:

- $\text{BI} = \frac{\text{BLKSIZ}}{\text{L1}}$
- First try = $\frac{\text{PH1B1}}{\text{BI}}$ (largest multiple)
- Second try = $\frac{\text{RECBLK}}{\text{BI}}$ (new or original)

REDUCE, AK-H5

RECBLK is reduced by one and a branch is made back to STEP3A to try to optimize BPT.

The result of the first try is compared to that of the second try and the lesser value is stored in RECBLK.

The new RECBLK is now compared to PH1B2 and the greater value is stored as the effective RECBLK. The routine then continues at STEP4.

USEORIG, AK-J5

When RECBLK is not to be changed, original values are saved:

- OM (internal order of merge)
- B (number of records per internal sort block)
- SBSIZE (internal sort block size in bytes)
- BPT (maximum number of internal sort blocks per 2311 track)
- G (number of records per phase 1 sequence)

STEP4, AK-E4

The number of bytes per sort block (SBSIZE) is calculated:

$$\text{SBSIZE} = \text{RECBLK} \times \text{INTPRL}$$

* Rounded to the next lower number.

- GA1 (available core for sort)
- GAREA (number of bytes in a phase 1 sequence, excluding tag area, in GA1)
- GAREA1 (number of bytes in a phase 1 sequence, including tag area, in GA1)
- NOBLKG (number of sort blocks in GAREA)
- MXBYPT (maximum number of bytes per track)
- AMAX (size of work area)
- NOPASS (number of merge passes in phases 2 and 3).

A branch is then made to PUTCO2.

CBYPTK, AK-H3

The maximum number of bytes per track (MXBYPT) and the maximum work area size (AMAX) are calculated:

- $MXBYPT = SBSIZE \times BPT$
- $AMAX = 2[(FILESZ \times L1)/MXBYPT] + 3$

*Rounded to the next higher whole number.

If the CALCAREA option is specified (CALCSW2 switch on - Chart AL), a branch is made to INITCSWS. If not the CALCAREA option, AMAX is compared to the work area given by the user (TRACKS). If AMAX is greater, a branch is made to SW1; if AMAX is equal to or less than TRACKS, the branch is to COMPG.

SW1, AL-D2

The constants just calculated are saved but are not used. To accomplish this, the second byte of the OM is ORED with a hexadecimal F0 to signal future overlays that the calculated constants for this particular OM are nullified and cannot be used. (For example: an OM of 0007 is made 00F7.)

If RECBLK is greater than one, a branch is made to REDUCE (Chart AK) to recalculate constants for the current OM using a decreased RECBLK. The set of constants resulting from the recalculation will be substituted for the nullified set for the current OM.

If RECBLK is equal to one and the last set of constants has not been calculated

(OM>2), a branch is made to STEP1B (Chart AK) to recalculate all constants using a lower OM.

If RECBLK is equal to one and the last set of constants has been calculated (OM = 2), a branch is made to OUTPT.

COMPG, AL-D1

Constants GA1, GA2, GAREA, and GAREA1 are calculated; these will be used to compute G area.

- GA1 (available core for sort)

$GA1 = CORESZ - SUPER - PHS1 - LEN1 - SBSIZE$

CORESZ Number of bytes of core storage available for the sort program.

SUPER Size of supervisor.

PHS1 Phase 1 program size.

LEN1 Number of bytes required for user program in phase 1.

SBSIZE Sort block size (output area) in bytes.

Note: If blocking is variable, the overflow area must be equal to the minimum input block minus L1 (maximum number of bytes in a single or logical input record).

- GA2 (number of sort blocks, plus an 8-byte tag for each record in the sort block, in GA1).

$GA2 = \frac{GA1}{SBSIZE + [8(SBSIZE/L1)]} *$

L1 Maximum length of input record.

*Rounded to the next lower whole number.

Note: The divisor in this calculation is saved in SBSZPTAG and is used in later calculations.

- GAREA (number of bytes in a phase 1 sequence, excluding tag area, in GA1).

$GAREA = SBSIZE \times GA2$

- GAREA1 (number of bytes in a phase 1 sequence, including tag area, in GA1).

$GAREA1 = GA2 [SBSIZE + ((SBSIZE/SBSZPTAG) \times 8)]$

Figure 15 is an illustration of the sort area in core as determined by the calculations in COMPG.

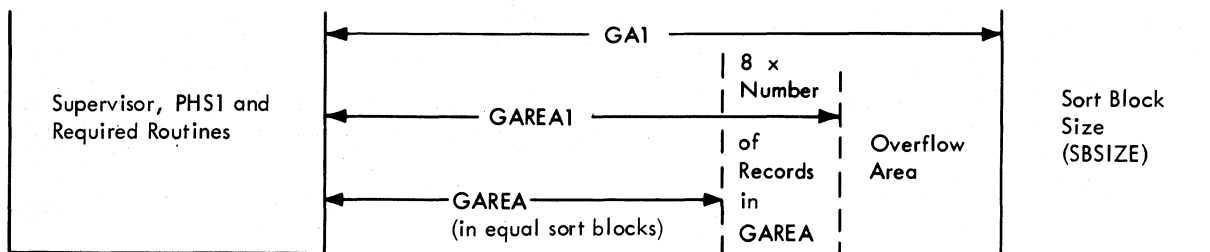


Figure 15. Sort Area Layout - Overlay 8

RETRY, AL-E1

For variable blocking, a branch is made to BYPVAR. For fixed blocking, a test is made to determine if GAREA is a multiple of the input block size (BLKSIZ). If it is, a branch is made to ENDGF to compute G; otherwise, the routine continues at BYPVAR.

BYPVAR, AL-G1

The size of the overflow area is calculated and if the size is adequate a branch is made to ENDGF to compute G. If the area is too small, GAREA1 is reduced by the quantity (SBSIZE + SBSZPTAG), and the routine branches back to RETRY until a valid overflow area is established. If no overflow area exists, a branch is made to SW1 to nullify current calculations.

ENDGF, AL-A3

Several constants are calculated and saved:

- Final G (number of records per phase 1 string).

$$G = \frac{GAREA}{L1}$$

- B (number of records per sort block).

$$B = \frac{SBSIZE}{L1}$$

- NOBLKG (number of sort blocks in G).

$$NOBLKG = \frac{GAREA}{SBSIZE}$$

- NOPASS (number of merge passes)

File size (FILESZ) is divided by G and rounded high to obtain the maximum number of sequences. The OM is then

multiplied by itself until a product equal to or greater than the number of sequences is reached. Each multiplication is counted, and the count is taken as the number of merge passes (NOPASS). If the number of sequences is less than the OM, NOPASS is made equal to one and phase 2 of the sort/merge program will be bypassed.

The routine then continues at PUTCON.

PUTCON, AL-C3

If the CALCAREA option is specified (switch PUTCONSW is on) and the first series of calculated constants for each OM have been saved, a branch is made to CALRPT. Otherwise, the final BPT for the current OM is calculated and constants are saved:

- OM (internal order of merge)
- B (number of records per internal sort block)
- SBSIZE (internal sort block size in bytes)
- BPT (maximum number of blocks per 2311 track)
- G (number of records per phase 1 sequence)
- GAI (available core for sort)
- GAREA (number of bytes in a phase 1 sequence, excluding tag area, in GAI)
- GAREA1 (number of bytes in a phase 1 sequence, including tag area, in GAI)
- NOBLKG (number of sort blocks in GAREA)
- MXBYPT (maximum number of bytes per track)
- AMAX (size of work area)

- NOPASS (number of merge passes in phases 2 and 3).

The routine then continues at CALRPT.

CALRPT, AL-A4

If switch CALRPT is on (unconditional branch), this routine is bypassed by branching to REINIT.

If the switch is off (no-op), switches CALRPT and PUTCONSW are turned on and a new value is calculated for records per track (RPT). The new RPT is compared to the previous RPT, and if the new value is greater, the new RPT and AMAX are saved and the routine continues to REINIT.

If the new RPT is less than the previous RPT, a branch is made to CKRECBK.

REINIT, AL-B4

Switches CALRPT and PUTCONSW are turned off (no-op) and a new RPT is calculated and compared as described in the preceding function block (CALRPT).

CKRECBK, AL-B5

If RECBLK is equal to one, switches PUTCONSW and CALRPT are turned off (at REINIT1) and the routine branches to PUTCO2.

If RECBLK is not equal to one and switch PUTCONSW is on, a branch is made to PUTCO2 to reduce OM and recalculate constants. If RECBLK is not equal to one and PUTCONSW is off, RECBLK is reduced by one and a branch is made back to STEP3A (Chart AK) to recalculate.

PUTCO2, AL-F4

If the last calculation has been made (OM = 2), a branch is made to OUTPT. When OM is greater than 4, OM is reduced by 1 and a branch is made to STEP1B (Chart AK) to recalculate all values. When OM is equal to or less than 4, the minimum sizes for phases 2 and 3 are used, OM is decreased by 1, and a branch is made to STEP1B (Chart AK) to recalculate all values.

OUTPT, AL-G3

- For tape input/output, this routine is bypassed to fetch the next overlay at FETCH9.
- For disk output, output BYTBLK, BYBKLS, and BPT3OP are calculated for phase 3.
- For disk input, input BYTBLK, BYBKLS, and BPT1IP are calculated for phase 1.

A branch is then made to FETCH9.

FETCH9, AL-H3

The next overlay (DSORT009) is fetched if no errors were detected. If there were any errors, the last overlay (DSORT010) is fetched.

COMPUTE CONSTANTS (VARIABLE-LENGTH RECORDS)
- AM

This routine (part of overlay 8) calculates constants for variable-length records for a sort run. Formulas used in the calculations are described in detail in IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.

VARRTN, AM-B1

For a merge-only run, a branch is made to fetch overlay 10 at FETCHEND. If CALCAREA option is not specified, a branch is made to STEP1V. For other than merge-only or CALCAREA runs, switches CALCSW1V and CALSW2V are turned on (unconditional branch) and the routine continues to STEP1V.

STEP1V, AM-D1

In order to compute the work area for variable-length records, it is necessary to calculate:

- MAXBL1 (maximum block size for phase 1)
- MAXBL2 (maximum block size for phase 2)
- MAXBL3 (maximum block size for phase 3)

$$\text{MAXBL1} = \frac{\text{CORESZ} - \text{SUPER} - \text{PHS1} - \text{LEN1} - \text{BLKSIZ}}{2 + (8/L4)}$$

$$\text{MAXBL2} = \frac{\text{CORESZ} - \text{SUPER} - \text{PHS2} - (\text{L1} \times \text{OM})}{\text{OM} + 1}$$

L1 length, in bytes, of largest input record.

OM Order of Merge (ranges from 2 to 6).

$$\text{MAXBL3} = \frac{\text{CORESZ} - \text{SUPER} - \text{PHS34} - \text{LEN34} - \text{BLKSOZ} - (\text{L1} \times \text{OM})}{\text{OM}}$$

PHS34 Phase 3 program size.

LEN34 Number of bytes required for user program in Phase 3.

BLKSOZ Length, in bytes, of a single output block.

If the calculated value of MAXBL2 or MAXBL3 exceeds 3624, the calculations are disregarded and 3624 is substituted.

MAXBL1 is compared to MAXBL2 and the smaller of the two values is then compared to MAXBL3. The smaller of the latter two values is used as the final MAXBL3 and, if less than 3624, it is also used as SBSIZE. If MAXBL3 is not less than 3624, SBSIZE is made equal to 3624. The routine then continues to COMBPTV.

COMBPTV, AM-G1

A branch and link is made to COMPBT+4 (Chart AK) to calculate the number of blocks per track. Upon return to this routine, the BPT just calculated is compared to the previous BPT. If the new BPT is greater, a branch is made to CBYPTKV. If the previous BPT is greater or equal, the routine continues to REDUCEV.

CBYPTKV, AM-J1

The maximum number of bytes per track (MXBYPT) is calculated by multiplying sort block size (SBSIZE) by number of sort blocks per track (BPT) and the routine continues to COMPGV.

COMPGV, AM-A4

G area constants are calculated and saved:

- GA1 (available core for sort)

$$\text{GA1} = \text{CORESZ} - (\text{SUPER} + \text{PHS1} + \text{LEN1} + \text{SBSIZE} + \text{BLKSIZ})$$

This calculation is similar to that for fixed length records and is described in detail in the narrative for COMPG (Chart AK).

- NOBLKG (number of sort blocks in GAREA)

$$\text{NOBLKG} = \frac{\text{GA1}}{[(\text{SBSIZE}/L4) \times 8] + \text{SBSIZE}}$$

L4 minimum length, in bytes, of single logical input record.

Note: BLKSEQ is stored in NOBLKG after calculation.

- G maximum (maximum records per sequence in Phase 1)

$$\text{G maximum} = \frac{\text{NOBLKG} \times \text{SBSIZE}}{L4}$$

Note: G maximum is stored in GAREA1.

- B (average number of records per block)

$$B = \frac{\text{SBSIZE}}{\text{LAVG}}$$

LAVG = L5 = the mean length of a logical input record (the intermediate value between L1 and L4, or L1).

- GAVG (number of records in a string/sequence)

$$\text{GAVG} = \frac{\text{SBSIZE} \times \text{NOBLKG}}{\text{LAVG}}$$

Store GAVG in G after calculation.

- GAREA (number of bytes in a phase 1 sequence, excluding tag area, in GA1)
GAREA = NOBLKG X SBSIZE

- AMAX (maximum number of work area tracks)

The first step in the calculation of AMAX is to divide FILESZ (number of input records) by GAVG (number of records per sequence). The quotient, rounded high if there is a remainder, is the number of sequences; this value is stored in NOSEQ. The second step is to divide by BPT. This quotient, rounded low if there is a remainder, is

the portion of a sequence that can be written on one 2311 track. The final step is to multiply the two quotients (BLKSEW divided by BPT) (NOSEQ), double the result, and add 3 to include checkpoint tracks.

$$\text{AMAX} = 2[(\text{NIR}/\text{RECSEQ}) * (\text{BLKSEQ}/\text{BPT})\#] + 3$$

NIR Number of input records given by user.

*rounded to next higher whole number
#rounded to next lower whole number

The routine then continues to CALSW2V.

CALSW2V, AM-C4

If the CALCAREA option is specified (switch CALSW2V is on), a branch is made to CALNOP; if not, the calculated work area (AMAX) is compared to the work area given by user (TRACK). If AMAX is equal to or less than TRACK, the constants calculated are optimum values for the current OM and a branch is made to CALNOP. If AMAX exceeds TRACK, all calculated constants are nullified. To accomplish this, the second byte of the OM is Ored with a hexadecimal F0 to signal future overlays that the constants for this particular OM cannot be used.

If SBSIZE is greater than L1, a branch is made to REDUCEV to initialize for recalculation of constants using a lower OM. If SBSIZE is not greater than L1, the routine continues to SW2V.

CALNOP, AM-C5

The number of merge passes (NOPASS) required for sort are calculated:

If the number of sequences is less than the current OM, NOPASS is made 1 and the routine branches to PUTCOV. If the number of sequences is equal to or greater than the current OM, the count (NOPASS) is started at 1 and the OM is multiplied by itself until the product equals the number of sequences (NOSEQ). NOPASS is incremented by 1 for each multiplication. The routine then branches to PUTCOV.

PUTCOV, AM-D5

A branch and link is made to COMBPT+4 (Chart AK) to calculate the final sort BPT. Upon return to this routine (at PUTCOV+8), the constants calculated for the current OM are saved:

- OM (internal order of merge)
- B (average number of records per internal sort block)
- SBSIZE (internal sort block size in bytes)
- BPT (maximum number of internal sort blocks per 2311 track)
- G (average number of records per phase 1 sequence)
- GA1 (available core for sort)
- GAREA (number of bytes in a phase 1 sequence, excluding tag area, in GA1)
- GAREA1 (maximum number of records in a phase 1 sequence)
- NOBLKG (number of sort blocks in GAREA)
- MXBYPT (maximum number of bytes per track)
- AMAX (calculated size of work area)
- NOPASS (number of merge passes in phases 2 and 3)

The routine then branches to CKAMAX to test for best AMAX.

CKAMAX, AM-E5

The calculated AMAX is compared with the previous AMAX (BESTBKT). The better of the two values is saved in BESTBKT and a branch is made to SW2V to continue calculations.

REDUCEV, AM-C3

The calculated MAXBL3 (equivalent to SBSIZE) is compared with L1 (maximum input record length). If MAXBL3 is greater than L1, it is decreased by the value of L1 and compared to L1 again. If MAXBL3 is now equal to or less than L1, a branch is made to SW2V. If MAXBL3 remains greater than L1, a branch is made back to COMBPTV to recalculate constants using the lower MAXBL3 value (SBSIZE).

If in the first comparison at REDUCEV MAXBL3 is found to be equal to or less than L1, a test is made for the CALCAREA option. If the option has been specified, a branch is made to SVORIG; if it has not been specified, the branch is to SW2V.

SVORIG, AM-E3

The original constants were found to be best for the current OM and are saved. These constants and their definitions are the same as those listed under PUTCOV.

The routine then continues to CKAMAX.

SW2V, AM-F4

If the last set of constants has been calculated (OM=2), a branch is made to FETCH9 to fetch the next overlay. If another set of constants has yet to be calculated (OM>2), the new save area address is calculated by adding 32 bytes to the address in general register R3. If the OM is equal to or less than 3, the minimum phase 2 and 3 sizes are used in the calculation of the next set of constants.

The OM is then reduced by one and a branch is made back to location PHZ2 in the listing (calculation of MAXBL2 in the narrative for STEP1V) to recalculate all constants for the new OM.

FETCH9, AM-H4

If no errors were detected, the next overlay (DSORT009) is fetched. If any errors were detected, overlay DSORT010 is fetched.

COMPUTE CONSTANTS FOR ADDRROUT SORT - AN

This routine (part of overlay 8) calculates constants for use with the RAF-type (ADDRROUT) sort for fixed or variable-length records. Formulas used are described in detail in IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.

58 IBM S/360 DOS Sort/Merge

RAFRTN, AN-B1

If the CALCAREA option is specified, switch CALSWR2 is turned on (unconditional branch). In any case, the routine continues to STEP1R.

STEP1R, AN-D1

Constants for use in the ADDRROUT run are calculated:

- PH1B1 (maximum number of records contained in a phase 1 sort block)

- $$PH1B1 = \frac{CORESZ - (PHS1 + BLKSIZ + SUPR + LEN1)}{2 [(CF+10) + 8]}$$

CORESZ number of bytes of core storage available for sort program (CORESZ)

PHS1 Phase 1 program size

BLKSIZ input block length

SUPER supervisor size

LEN1 Number of bytes required for user program in phase 1

- CFPL10 (calculated control field length+10)

$$CFPL10 = TLACFD + 10$$

TLACFD total length of all control fields

- PH1B1 is recalculated using the formula:

$$PH1B1 = \frac{3624}{CFPL10}$$

The results of the two calculations for PH1B1 are compared and the lesser value is stored as the effective PH1B1. The routine then continues to STEP1BR.

STEP1BR, AN-E1

Additional constants are calculated:

- $$BL2 = \frac{CORESZ - (SUPER + PHS2)}{OM + 1}$$

- MAXBL2 (maximum number of records per phase 2 block)

$$\text{MAXBL2} = \frac{\text{BL2}}{\text{CFPL10}}$$

MAXBL2 is compared to PH1B1 and the lesser value is saved in RECBLK.

- $\text{BL3} = \frac{\text{CORESZ} - (\text{PHS34} + \text{BLKSOZ} + \text{TLACFD} + \text{SUPER} + \text{LEN34})}{\text{OM}}$

- MAXBL3 (maximum number of records per phase 3 block)

$$\text{MAXBL3} = \frac{\text{BL3}}{\text{CFPL10}}$$

PHS34 - Computed phase 3 size for ADDR0UT

BLKSOZ - Output block size

TLACFD - Total length of all control fields

LEN34 - length of user program in phase 3.

MAXBL3, MAXBL2, and PH1B1 are compared to RECBLK and the lesser value is stored as the maximum number of records per block (RECBLK).

STEP4R, AN-G1

The sort block size (SBSIZE) and maximum number of blocks per 2311 track (BPT) are calculated:

- $\text{SBSIZE} = \text{RECBLK} \times \text{CFPL10}$
- BPT - A branch and link is made to COMBPT+4 (Chart AK) to calculate the blocks per track.

Upon return to this routine (at STEP4R+24) the newly-calculated BPT is compared to the previous BPT (SAVE+6). If the new BPT is greater, a branch is made to CBYPTRK to continue calculations. If the previous BPT is greater or equal, the routine continues to REDUCER.

REDUCER, AN-J1

If the CALCAREA option is specified (switch CALSWR1 is on) a branch is made to USEORIGR. If the CALCAREA option is not specified and RECBLK is equal to one, the previous BPT is the optimum value for this set of calculations; a branch is made to SW1R to nullify the calculations just made and to recalculate all values using a lower OM.

If RECBLK is greater than one, it is decreased by one and a branch is made back to STEP4R to recalculate SBSIZE and BPT.

USEORIGR, AN-J2

The original values that were passed to overlay 8 are saved:

- OM (internal order of merge)
- B (number of records per internal sort block)
- SBSIZE (internal sort block size in bytes)
- BPT (maximum number of internal sort blocks per 2311 track)
- G (number of records per phase 1 sequence)
- GA1 (available core for sort)
- GAREA (number of bytes in a phase 1 sequence, excluding tag area, in GA1)
- GAREA1 (number of bytes in a phase 1 sequence, including tag area, in GA1)
- NOBLKG (number of sort blocks in GAREA)
- MXBYPT (maximum number of bytes per track)
- AMAX (size of work area)
- NOPASS (number of merge passes in phases 2 and 3)

A branch is then made to PUTCO2R.

CKBYPTRK, AN-A5

The maximum number of bytes per track (RPT) and the maximum work area (AMAX) are calculated and stored:

- $\text{RPT} = (\text{RECBLK} \times \text{BPT})$
- $\text{AMAX} = 2 (\text{FILESZ}/\text{RPT}) + 3$

In the AMAX calculation, FILESZ is the number of records to be sorted, as given by the user. Three is added to include the checkpoint tracks.

If the CALCAREA option is not specified (switch CALSWR2 is off) AMAX is compared to the work area given by the user (TRACK). If TRACK is equal to or greater than AMAX,

a branch is made to COMPGR. If AMAX is greater than TRACK, the routine continues to SW1R to nullify the calculations just completed.

If the CALCAREA option is specified (switch CALSWR2 is on), a branch is made to INCSWR where switches CALSWR1 and CALCSW3R are turned on, after which a branch is made to COMPGR.

COMPGR, AN-B4

G area constants are calculated:

- GA1 (available core for sort)

$$GA1 = CORESZ - (PHS1+SBSIZE+BLKSIZ+SUPER+LEN1)$$

- GA2 (number of sort blocks, plus an 8-byte tag for each record in the sort block, in GA1)

$$GA2 = \frac{GA1}{SBSIZE+8(SBSIZE/CFPL10)}$$

Note: The value of the denominator used in this calculation is stored in SBSZPTAG for later use.

- GAREA (number of bytes in a phase 1 sequence, excluding tag area, in GA1)
GAREA = GA2 X SBSIZE
- GAREA1 (number of bytes in a phase 1 sequence, including tag area, in GA1)
GAREA1 = SBSZPTAG X GA2

ENDGFR, AN-C4

Additional constants are calculated for the ADDRROUT run:

- Final G (number of records per phase 1 sequence)

$$G = \frac{GAREA}{CFPL10}$$

- B (number of records per sort block)

$$B = \frac{SBSIZE}{CFPL10}$$

- NOBLKG (number of sort blocks in G)

$$NOBLKG = \frac{GAREA}{SBSIZE}$$

- NOPASS (number of merge passes)

File size (FILESZ) is divided by G and rounded high to obtain the maximum number of sequences. The OM is then multiplied by itself until a product equal to or greater than the number of sequences is reached. Each multiplication is counted, and the count is taken as the number of merge passes (NOPASS). If the number of sequences is less than the OM, NOPASS is made equal to one and phase 2 of the sort/merge program will be bypassed.

The routine then branches to PUTCNSWR.

PUTCNSWR, AN-D4

If the CALCAREA option is specified (switch PUTCNSWR is on), a branch is made to CKRPT. If the CALCAREA option is not specified (switch PUTCNSWR is off), a branch and link is made to COMBPT+4 (Chart AK) to calculate the maximum number of blocks per 2311 track (BPT). Upon return to this routine (at PUTCNSWR+8) the calculated constants are saved:

- OM (internal order of merge)
- B (number of records per internal sort block)
- SBSIZE (internal sort block size in bytes)
- BPT (maximum number of internal sort blocks per 2311 track)
- G (number of records per phase 1 sequence)
- GA1 (available core for sort)
- GAREA (number of bytes in a phase 1 sequence, excluding tag area, in GA1)
- GAREA1 (number of bytes in a phase 1 sequence, including tag area, in GA1)
- NOBLKG (number of sort blocks in GAREA)
- MXBYPT (maximum number of bytes per track)
- AMAX (size of work area)
- NOPASS (number of merge passes in phases 2 and 3)

CKRPT, AN-A2

If switch CKRPT is on (unconditional branch), it and switch PUTCNSWR are turned off and a branch is made to CKRPT1. If switch CKRPT is off (no-op), it and switch PUTCNSWR are turned on and the routine continues to CKRPT1.

CKRPT1, AN-C2

The new RPT (number of records per 2311 track) is compared with the previous RPT. If the previous value is greater, it and AMAX are saved as the optimum values calculated up to this point and a branch is made to CKRECBKR. If the new value is greater, a branch is made directly to CKRECBKR.

CKRECBKR, AN-D2

If RECBLK (number of records per block) is equal to one, switches CKRPT and PUTCNSWR are turned off and the routine branches to PUTCO2R.

If RECBLK is not equal to one and switch PUTCNSWR is off (no CALCAREA option), a branch is made to PUTCO2R to reduce OM and recalculate constants. If RECBLK is not equal to one and switch PUTCNSWR switch is on (CALCAREA option specified), RECBLK is decreased by one and a branch is made to STEP4R to recalculate.

PUTCO2R, AN-F3

A test is made to determine if the calculation just completed is the last (OM=2). If it is, a branch is made to OUTPT (Chart AL). If the calculation just completed is not the last and OM is greater than 4, OM is reduced by one and a branch is made to STEP1R to recalculate all values. When OM is equal to or less than 4, the minimum sizes for phases 2 and 3 are used, OM is reduced by one, and a branch is made to STEP1R to recalculate all values.

SW1R, AN-D5

The constants just calculated are saved but are not used. To accomplish this, the second byte of the OM is Ored with a

hexadecimal 0F (for example, and OM of 07 is made F7) to signal future overlays that the calculated constants for this particular OM are nullified and cannot be used.

If RECBLK is greater than one, a branch is made to REDUCER to recalculate constants for the current OM using a decreased RECBLK. This set of recalculated constants will be substituted for the nullified constants for the current OM.

If RECBLK is equal to one and the last set of constants has not been calculated (OM>2), the constants are to be recalculated using a lower OM. The OM is reduced by one, (as described under PUTCO2R) and a branch is made to STEP1R.

If RECBLK is equal to one and the last set of constants has been calculated (OM=2), a branch is made to OUTPT (Chart AL).

SELECT ORDER OF MERGE - AP, AQ

This routine (overlay 9) selects the order of merge that produces the most efficient sort run on a time basis. The overlay utilizes:

- Record length.
- Input/output blocking.
- User-given file size.
- System model type.
- Available I/O and disk work areas.

Sort times are calculated for each possible order of merge. Each sort time consists of:

- Machine process time.
- Disk read and write time.
- Seek time.

Not all orders of merge are possible with any sort run because of disk work area limitations.

DSORT009, AP-B1

Base registers are initialized and a branch is made to BEGIN.

BEGIN, AP-C1

The output block size (IBOC) is obtained from BLKSOZ and a test is made for merge-only run. If this is a merge-only run, the entire routine for selecting order of merge and calculating sort time is not required; a branch is made to FETC10 to fetch overlay 10.

For a sort run, the input block size (IBIC) and sort record length (IRL) are obtained from BLKSIZ and INTPRL, respectively. For fixed-length records or an ADDRROUT run, a branch is made to FIXED. For variable-length records, IRL is re-initialized with L5 (average length of input records) before branching to FIXED.

FIXED, AP-H1

Constants and registers are initialized:

- MS (machine size) with CORESZ
- LOWTM (lowest sort time) with LMAXNO (7FFFFFFF).
- Registers 6 and 10 with zeroes.
- SPDFAC (speed factor) with value according to machine model
- FIOCS (time to execute IOCS) with value according to machine model

The input blocking factor (BI) is then calculated and stored:

$$BI = \frac{(IBIC) \text{ input block size}}{(IRL) \text{ input record length}}$$

For an ADDRROUT sort, IRL is re-initialized to contain the total length of control fields (TLACFD) plus 10. The routine then continues to RETRY.

RETRY, AP-J1

The trial order of merge is retrieved from SAVE table and tested for validity. If invalid, a branch is made to FINAL. For a valid order of merge, IBR (records in a sort block), G (size of internal sort sequence), and BPT (sort blocks per track) are obtained from the SAVE table. Then IRPT (records per track), NRTHWK (number of tracks required for half of work area), IBC (characters in a sort block), and BO (output blocking factor) are calculated and

stored. The routine then continues to STBO.

STBO, AP-D2

For tape input or for variable-length records, the routine branches to CLUB.

For disk input and fixed-length records, the number of tracks necessary for the input (NRTIN) is calculated before continuing at CLUB:

$$NRTIN = \frac{(BPT1IP) \times (BI)}{V} *$$

*Rounded high by adding 1 if any remainder.

BPT1IP Input blocks per track.
 BI Input blocking factor.
 V User given file size (number of records to be sorted -- FILESZ).

CLUB, AP-G2

For disk output with either fixed-length records or ADDRROUT, a branch is made to TAGOUT. For tape output or variable-length records, the calculations at TAGOUT are bypassed by a branch to SPADE.

TAGOUT, AP-K2

The number tracks necessary for the output (NRTOUT) are calculated:

$$NRTOUT = \frac{(BPT3OP) \times (BO)}{V} *$$

*Rounded high by adding 2 if any remainder.

BPT3OP Output blocks per track.
 BO Output blocking factor.
 V FILESZ

SPADE, AP-A3

NRTOUT is stored and registers are initialized to estimate the total number of drives and tracks available for sort.

REP1, AP-B3

Table DEVTAB is prepared, using device numbers from the PUB table. DEVTAB contains a maximum of seven 4-byte entries. The first two bytes of each entry contain a device number obtained from TABLEB. The second two bytes contain the total number of tracks available on the device.

Note: TABLEB can contain more than one work area specified for the same device. DEVTAB accumulates the number of available tracks so that there is only one entry per device.

REPB, AP-C3

The number of tracks for the work area on the first drive is accumulated and stored in DEVTAB+2. This function is repeated, incrementing the DEVTAB location, for each available drive.

The number of tracks used for input and output are estimated and the estimated total number of tracks used by the sort is accumulated and stored in NRTRKS.

B118, AP-D3

The number of sequences (strings) produced by phase 1 is calculated and stored in STRING:

$$\text{STRING} = \frac{V}{G} *$$

*Rounded high by adding 1 if any remainder.

V FILESZ (number of records to be sorted).

G Number of records in a phase 1 string.

If V (number of records to be sorted) is less than G (number of records in a phase 1 string), the value in LOWTM is reduced to indicate this condition and a branch is made to VLESSG. If V is not less than G, the routine branches to STVSOM.

STVSOM, AP-G3

If the number of phase 1 strings (STRING) is less than the order of merge (OM), a branch is made to FINAL; if not, the routine continues to B119.

B119, AP-H3

The time required in phase 1 to read input is calculated.

B126, AP-J3

SKT2 is accumulated for use in the calculation of phase 2 seek time. (See TABLEF function, Chart AQ.) B126 is repeated until the last pass, when the routine continues to B135.

B135, AP-A4

WRT (write time for phase 1, read and write time for phases 2 and 3) is calculated.

B136, AP-B4

The assumed limits for disk work area are determined and stored in LL1, LL2, LL3, and LL4.

B170, AP-C4

Phase 2 seek time is calculated, using SKT2 calculated at B126. Phase 2 process time is then calculated and added to SKT2.

E181, AP-D4

Phase 3 seek time (SKT3) is calculated. (See TABLEF function, Chart AQ.)

S188, AP-E4.

Phase 1 seek, process, and format time is calculated and accumulated as a sum of its components: RDT1 (phase 1 read time), WRT (calculated at B135), SKT2 (calculated at B170), and SKT3 (phase 3 seek time).

E200, AP-F4

The lowest sort time (LOWTM) is determined by comparing the time just calculated to the previous calculated time. The lowest value is saved in LOWTM.

VLESSG, AP-G4

The contents of register ROM are saved in SAVROM (points to the parameters in SAVE table from which the time just calculated has been derived).

FINAL, AP-H4

When calculations are completed for the last order of merge, a branch is made to RESTOR; otherwise, the routine returns to RETRY.

RESTOR, AP-H5

The maximum file size (MFS) is calculated and stored. If a sort time has been determined, the appropriate SAVE table contents (determined by pointer in SAVROM) are moved to the respective location in the PHASE table.

If no time has been determined, an error message is printed.

If the CALCAREA option is specified, the optimum number of work area tracks is stored in TRACK.

FETC10, AP-C5

Overlay 10 is fetched.

TABLEF, AQ-B2

The TABLEF subroutine is used in the calculation of the seek time for each pass and for phase 3. A search of three tables (TABLE, TABLE2, and TABLE3) is made and, for each table a result (ANS1, ANS2, ANS3) is calculated using OM (order of merge), BPT (sort blocks per track), and CPST (number of cylinders per string).

SEEKTM, AQ-G5

The time for one seek (SKT) is calculated using various seek lengths (SKLEN), given the number of cylinders per seek.

- SKLEN ≥ 26

$$SKT' = \frac{(SKLEN-26) \times 6800}{177} + 7700$$

- SKLEN < 26 but ≥ 4

$$SKT = [((SKLEN-4) \times 2100) / 22] + 5600 + SKT'$$

- SKLEN < 4

$$SKT = [(SKLEN \times 2600) / 4] + 3000 + SKT'$$

PRINT CONTROL CARD AND FETCH NEXT PHASE - AR, AS

When the PRINT option is specified in the SORT control card, this routine (overlay 10) extracts and prints the information contained in the sort control cards. The constants that were calculated by the assignment phase are also extracted (from the phase table) and printed. The routine then checks for errors and type of run (sort or merge), and fetches the required phase 1, 2, or 4 overlay.

If errors are detected, messages are printed and, depending on the unit assignments explained in overlay 1 (Chart AA), the program either provides for operator correction of the error and job re-run, or cancels the job.

For a merge-only run, the checkpoint record (phase tables) is written on disk and overlay DSORT401 of phase 4 is fetched.

For a RESTART run (available for sort only) the old checkpoint record is read and tested and one of four phase 2 overlays is fetched:

- ADDROUT/Fixed-length records, OM 2-4....DSORT201

- ADDROUT/Fixed-length records,
OM 5-7....DSORT202
- Variable-length records,
OM 2-3....DSORT203
- Variable-length records,
OM 4-6....DSORT204

An invalid checkpoint record is indicated by one of two error messages:

- Invalid Restart (7D53D): indicates sort run was terminated before checkpoint was written by phase 1.
- Invalid Restart (7D55A): indicates incorrect disk pack placement.

For a sort run, the sort checkpoint record (phase tables) is written, and phase 1 overlay DSORT101 is fetched.

DSORT010, AR-B1

The base registers are initialized and a branch is made to CKPRINT.

CKPRINT, AR-C1

For a CALCAREA run, or for any other type run in which the PRINT option is specified, a branch is made to CCDESC. If neither the CALCAREA nor the PRINT option is specified, a branch is made to EOJ.

CCDESC, AR-E1

The job name and date are printed and a branch-and-link is made to PRNTCARD if the PRINT option is specified. If the PRINT option is not specified, a branch is made to TSTCALCS.

PRNTCARD, AR-G1

The print operation is initialized and a branch and link is made to PRERR, located in overlay 1, to print the heading. All the control card images are then printed (by successive BAL's to PRERR), after which the routine links back to TSTCALCS.

TSTCALCS, AR-J1

If the CALCAREA option is not specified, a branch is made to NOCALCS; for a CALCAREA run, the branch is to PRTCALCS.

PRTCALCS, AR-K1

This routine extracts and prints the results of calculations made by the program for:

- MINTRK - minimum number of work area tracks
- TRACK - optimum number of tracks for work area
- NOPASS - optimum number of passes

If this routine was entered as the result of the CALCAREA option and the PRINT option was not specified, a branch is made to EOJCALC. If the PRINT option was specified, the switch at EOJCALSW is turned on (unconditional branch) and a branch is made to NOCALCS.

NOCALCS, AR-C2

A test is made to determine that at least one file is specified. For a merge-only run, the sort printer routine (BRCH) is modified at location M1. The routine then continues at BRCH.

BRCH, AR-E2

The values given by the user (control fields, format code, size of sort, files) are extracted from the phase table starting at CF1LNG. The values are printed out (by successive BAL's to PRERR) and the routine continues at RE1.

RE1, AR-F2

The record type and record lengths are extracted from the phase table and printed. If only one length is given by user, an assumed L3 value is printed. For variable-length records, the values for L1 through L5 are printed; for fixed-length records, only L1 and L3 are printed.

INF, AR-G2

The input block size, block type (or assumptions, for variable-length records), input media, volume(s), and the CLOSE and OPEN options are extracted and printed.

OUTF2, AR-H2

The output block size, output media, and output OPEN and CLOSE options are extracted and printed. Then, if required, the user file name, address of user routine, and exit numbers for each phase are printed.

OPFLE, AR-K2

All other options given by the user, whether in the OPTION control card or other control cards, are extracted and printed.

CONOUT, AR-A4

Calculated constants are extracted and printed:

- Computed maximum file size for sort.
- Computed maximum input and output block sizes for sort/merge.
- User-given number of tracks for sort/merge.
- Sort/merge program origin.

Note: If the assignment phase was terminated before all of the calculations were completed, and if the PRINT option is specified, some constants will be zero or will not be printed.

EOJ, AR-B4

If the switch at EOJCALSW is on, indicating that the PRINT option was specified for a CALCAREA run, a branch is made to EOJCALC. With switch EOJCALSW off, the routine branches to EOJZ for a merge-only run. For a sort, the sort block size (SBSIZE) is increased by one for phase 2 and, if the number of passes (NOPASS) is odd, the pointers to the work area are reversed in an attempt to improve the sort time and

save a pass. The routine then continues to EOJZ.

EOJCALC, AR-C3

This routine is entered upon completion of the assignment phase for a CALCAREA run. Register 9 is initialized with the address of message E80I. Switch EOJCSW is turned on (unconditional branch) to provide the correct exit after the end-of-phase message is printed. A branch is then made to MSG91A (Chart AS) to print end-of-assignment-phase message for the CALCAREA run.

EOJZ, AR-E4

The checkpoint record number is updated, and locations NOBLKG+2 and NOBLKG+3 are initialized with output file label data and SYSLOG data, respectively. If no errors are detected at this point, the routine continues at TSMERG. If errors are detected, a branch is made to ABORT in overlay 1 (Chart AA).

TSMERG, AR-G4

For a merge-only run, a branch is made to WTCKMG; for a sort, the routine continues at TSREST (Chart AS).

WTCKMG, AR-H4

The checkpoint record is written for the merge-only run. If the PRINT option is specified, switch F4SW is turned on (unconditional branch) to provide the correct exit after the end-of-phase message is printed; a branch is then made to MSG91 (Chart AS).

If the PRINT option is not specified, a branch is made to FETCH4 to fetch the first overlay of phase 4.

TSREST, AS-B5

If this is not a RESTART run, a branch is made to WTCKPS; if a RESTART run, a branch-and-link is made to RDCKPR to read the previous checkpoint record. The

validity of the record is then tested and, if invalid, a branch is made to EOJ3RT.

If the previous checkpoint record is valid, a test is made for the PRINT option. If the option is specified, switch F2SW is turned on (unconditional branch) to provide the correct exit after the end-of-assignment-phase message is printed, and a branch is made to MSG91. If the PRINT option is not specified a branch is made directly to FETCH2.

WTCKPS, AS-B4

The checkpoint record for the sort run is updated and written on the specified device. If PRINT option is specified, a branch is made to MSG91 to initialize and to print the end-of-assignment-phase message. If the PRINT option is not specified, a branch is made directly to FETCH1.

EOJ3RT, AS-E5

The invalid-restart message is printed by a branch-and-link to ERROR in overlay 1. Then, depending on the unit assignments, the user may have the option to resume the run. If the user reply is IGNORE, the program will be re-entered at WTCKPS for a complete sort run; otherwise, the job is canceled.

MSG91, AS-C1

This location is entered at the end of the assignment phase, if the PRINT option is specified, for merge-only, sort, or restart runs. Register 9 is initialized with the address of message E91 and a branch is made to MSG91A.

MSG91A, AS-D1

The end-of-assignment-phase message (E91 for sort, merge-only, or restart, E80I for CALCAREA) is printed. The exit switches

that were initialized earlier in the phase are now tested. The exit for each type of run, in the order tested, is:

Switch on:	Go to:	Run type:
EOJCSW	EOJC1	CALCAREA
F4SW	FETCH4	MERGE
F2SW	FETCH2	RESTART
none	FETCH1	SORT

EOJC1, AS-F1

This is the end-of-job for a CALCAREA run. If no errors are detected, the EOJ macro is issued. If errors are detected, a branch is made to ABORT in overlay 1 (Chart AA).

FETCH4, AS-G2

The first overlay of phase 4 (DSORT401) is fetched for a merge-only run.

FETCH2, AS-F3

The appropriate phase 2 overlay is fetched, depending on the type of run for RESTART.

- DSORT201 for ADDRROUT or fixed-length records, order of merge 2-4
- DSORT202 for ADDRROUT or fixed-length records, order of merge 5-7
- DSORT203 for variable-length records, order of merge 2-3
- DSORT204 for variable-length records, order of merge 4-6

FETCH1, AS-E4

The first overlay of phase 1 (DSORT101) is fetched for a sort run.

Phase 1 initiates the sort operation. A modified internal merging technique sorts the input records into sequences and moves them to the disk work area.

Phase 1 is divided into five overlays:

- Overlay 1(DSORT101) - Initialization of the multi-volume (Exit 11 linkage) routine.
- Overlay 2 (DSORT102) - Read checkpoint and format-disk routine.
- Overlay 3 (DSORT103) - Initialization of input-data, internal-sort, and output-data routines for disk or tape input.
- Overlay 4 (DSORT104) - Initialization of input data, internal-sort, and output-data routines for an ADDRROUT type run.
- Overlay 5 (DSORT105) - End-of-phase routine.

The initialization routine modifies the sort program to perform a specific sort based on control information supplied by the user and processed by the assignment phase. It also:

- Initializes compare instructions.
- Defines the input/output record areas.
- Relocates and/or deletes several routines so that the program occupies the least amount of main storage for each type of sort run.
- Formats the tracks in the disk work area, based on the sort-block length.

Phase 1 reads records from disk or tape into main storage until the input area (G) is filled. Each record can be located by the address of its leftmost byte (Figure 16, section 1).

During the sort operation, the records remain in their original positions in the input area; only the record addresses are sorted in tables to indicate the order in which the records are to be moved to the output area.

There are basically two levels of sorting in phase 1. Level 1 builds sequences of two records each (doublets) from the records in G area. Two records at a time are compared to each other for high, equal, or low. The record selected is determined by the user-given sequence for each field (ascending or descending). The correctly-sequenced doublet is represented in a table by the addresses of the two records.

Level 2 merges the 2-record sequences into 4-record sequences in a second table. Again, the sequences are represented by the addresses of the records. The 4-record sequences are then merged into 8-record sequences in the first table. The merging continues between the two address tables until there are but two sequences left in either one of the tables (Figure 16, section 2). The two remaining sequences are then merged with the records being moved from the input area to the output area.

The records are moved from the input area to the output area in the order in which their addresses appear in the address table (Figure 16, section 3). The output area can be either equal to G or a sub-multiple of G so that the sequence may be written in a blocked format on disk. As the output sort-block is filled, it is written in the disk work area at its calculated interleaved address. Interleaving is an arrangement of the disk work area, determined by the assignment phase, that minimizes disk seek-time in phase 2 and 3.

After the sequence has been written on disk, the input volume or file is checked and, if more records are to be read, the input area (G) is refilled and the program returns to level 1.

If all records in the input file(s) have been processed into sequences, the end of phase routine is executed. Constants are calculated and tables are updated for phase 2.

Figure 17 is a main-storage layout of phase 1 showing the five overlays.

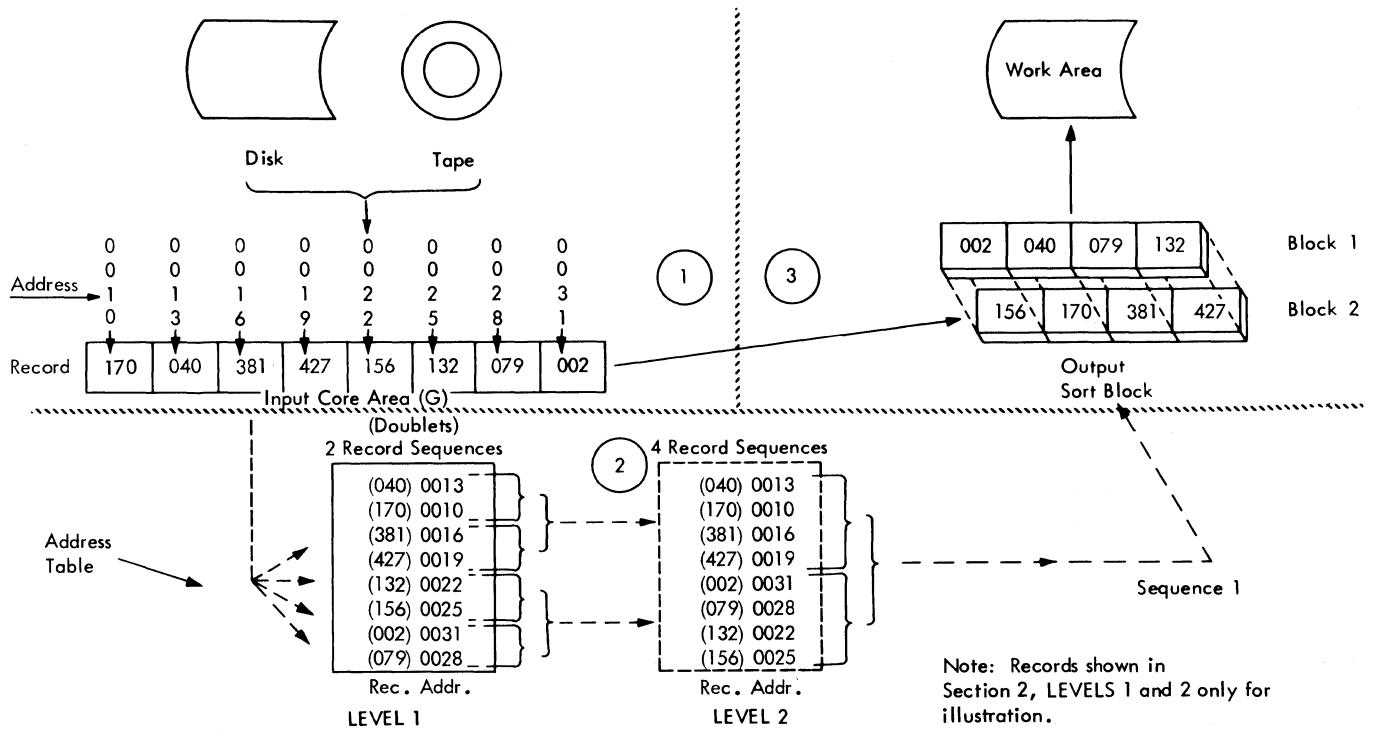
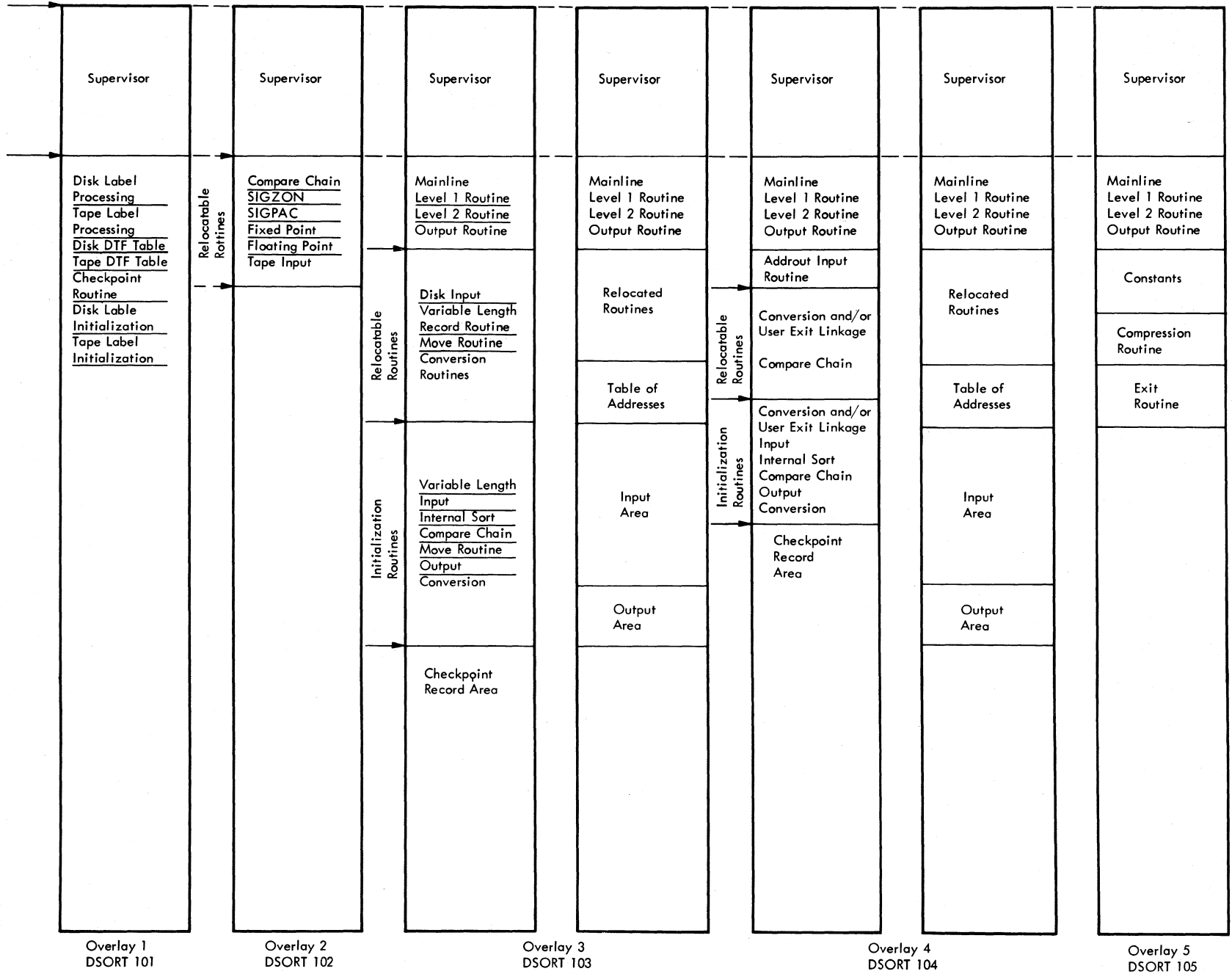


Figure 16. Phase 1 Internal Sorting (Ascending Sequence)

Figure 17. Phase 1 Main Storage Layout



Note: Not drawn to scale.

INITIALIZATION ROUTINE FOR MULTI-VOLUME
(EXIT 11 LINKAGE) - BA

The multi-volume - exit 11 linkage is initialized according to the specifications set by the user in the sort control cards.

The specifications include:

- File type (tape or disk input)
- Label type
 1. Disk - standard labels with or without user additional labels
 2. Tape - standard labels with or without user additional labels, non-standard labels, or unlabeled
- User options
 1. Alternate drive for tape input
 2. Rewind option for "open" or "close" times

When the multi-volume (exit 11 linkage) is initialized, it is written on disk in the checkpoint track and the second overlay of the phase (DSORT102) is fetched.

The format routine arranges the disk work area, cylinder by cylinder, for use with the interleaving technique. The address for each area (sort block) on disk is built from constants supplied by the assignment phase. The lower four bytes are based on the sort block length (Figure 18). The next four bytes (CCHH) are based on the lower limit of work area extents.

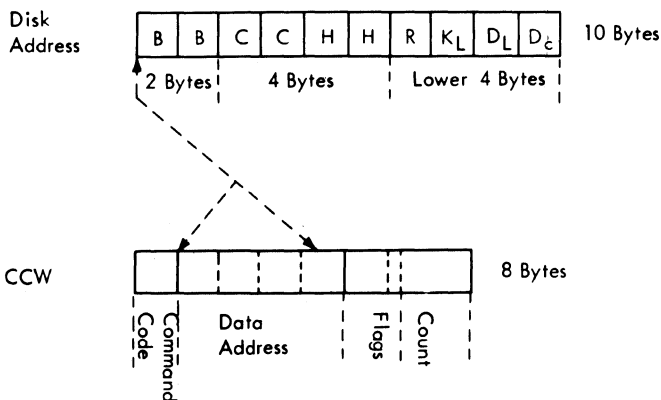


Figure 18. Disk Address Format

The disk work area format is started, using the CCW's that were built for a cylinder from the disk addresses just

completed. When the cylinder limits are reached, the CCHH bytes of the disk address are updated, and the process is repeated until the upper limit of the work area is reached.

INITEX11, BA-B1

The label routine base register is initialized and a branch is made to LBLINIT.

LBLINIT, BA-C1

The address of the checkpoint track and the user routine are obtained from the assignment phase table. Then, if the input file is on tape, a branch is made to INTAPE; if on disk, the routine continues to DSKINIT.

DSKINIT, BA-D3

The DTF table DSKDTF is initialized for disk operation and for user exit 11 (if specified). The routine continues at EOFINIT.

EOFINIT, BA-F3

The input file name and the EOF (end of file) routine address are inserted in the disk DTF table; a branch is then made to CHKPT.

INTAPE, BA-C4

The tape input switch (MNLDSK) is turned on and the DTF table TAPDTF is initialized for tape operation. The multi-volume routine and the DTF table are initialized for type of labels (standard, non-standard, or unlabeled) and for user exit 11, if specified. The routine then continues to INIT32.

INIT32, BA-H5

The tape DTF table is initialized for the open and close rewind options and a branch is made to CHKPT.

CHKPT, BA-G3

The second checkpoint track is formatted and the initialized multi/volume routine is written. The next overlay of phase 1 (DSORT102) is then fetched.

FORMAT ROUTINE - BB

The checkpoint record that was passed on from the assignment phase is read into main storage. Some of the data contained in the checkpoint record is then used to format the work area:

- Work area extents (TABLEB from the assignment phase)
- Blocks per track (BPT)
- Sort block length (SBSIZE)

IBV021, BB-B2

A channel program is initialized and the checkpoint record is read. The routine then continues at FORMAT.

FORMAT, BB-C2

For an ADDRUT run, the branch at FETCH is initialized so that overlay 4 will be fetched. Constants BPT, SBSIZE, and TABLEB are obtained from the checkpoint record and inserted in FRMBPT, register XWORK1, and WKTAB, respectively. If BPT is equal to 1, the format routine is bypassed by a branch to FETCH. If the BPT is 2 or more, CCW's are built to format a cylinder.

FORMGO, BB-F4

After the format for a cylinder has been established, new disk addresses are placed in existing CCW's to organize the next cylinder. Processing continues until the format of the entire work area is completed. A branch is then made to FETCH.

FETCH, BB-G4

Overlay 4 (DSORT104) is fetched for an ADDRUT run; overlay 3 (DSORT103) is fetched for all other runs.

INITIALIZATION FOR DISK OR TAPE INPUT - BC

After the disk work area has been formatted, this overlay (DSORT103) reads the checkpoint record and initializes the:

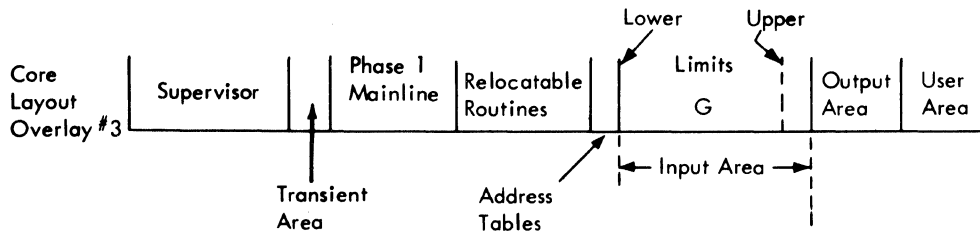
- Input routine.
- Internal-sort routine.
- Output routine.
- End-of-phase routine.

Input routine. The initialization routine determines which input routine is used for a sort run. Tape input routines are relocated to the area of the disk input routine.

Internal-sort routine. Several constants and addresses are initialized:

- G (number of records to be sorted).
- Number of sort blocks in G.
- Record length.
- Location of address table.
- Number of doublets.

The compare chain sub-routine is initialized; it is then relocated regardless of the input type. The lower and upper limits of the main-storage input area (Figure 19) are calculated and stored, as is the start of output area.



Note: Not drawn to scale.

Figure 19. Limits of Input Area

Output routine. The seek and search CCW's and CCB, and the DSHIFT table are initialized.

IBVAA1, BC-B3

G, number of sort blocks in G, and record length (computed by the assignment phase) are relocated for use by the internal sort routine. The number of doublets is computed and saved in NRDUB.

IBV022, BC-B2

The checkpoint record is read in at the end of the overlay and the routine continues at VLRINZ.

Constants supplied by the assignment phase are used to initialize the compare chain with the number of control fields and the displacement and length of each field. The compare string is relocated.

VLRINZ, BC-C2

The variable-length address routine is initialized if one or more of the following routines are specified:

If data conversion is specified, the conversion routine is relocated and initialized with the number of control fields and the location of the compare chain.

- Variable-length input.
- Data conversion.
- User exit 12.

FLMVGO, BC-F3

The variable-length address routine is also relocated, and the compare chain is initialized to load addresses from address table 1.

For fixed-length input, the move routine is initialized with the record length.

If user exit 12 is specified, the linkage to the user programming is prepared. The variable-length output routine is then initialized and relocated and the routine continues to INPTIZ.

FLMVBC, BC-G3

Using G supplied by assignment phase, calculations are made to determine the areas necessary for the address tables, the start and end addresses of the input area, and the start address of the output area.

INPTIZ, BC-G2

OUTIZ, BC-J3

A test is made to determine the type of input (disk or tape). Disk input is initialized to process key, if any; tape input is initialized and the sub-routine is relocated at P1INBG. The routine then continues at IBVAA1.

The output routine is initialized for interleaving. The disk address calculation routine is initialized to compute disk addresses for the sort run. The CCW and CCB are initialized, and the starting addresses for the first order of merge are calculated and placed in the DSHIFT table.

If variable-length records have been specified, the variable-length output routine is relocated to IBVMC3 and is initialized with the start address and upper limit of the output area.

A branch is then made to DINMVA (Chart BD) to write the mainline in the checkpoint track area.

INPUT ROUTINE FOR DISK OR TAPE - BD

The initialization routine of phase 1 determines which input routine (disk or tape) is to be used for any specific run. In either case, a test is made for a full G area. The input area consists of G plus an overflow area that is used when the input blocks or records exceed the G area. When the G area is full, the current address within the input area is stored and the program branches to the sort routine.

Disk or tape files were opened by the multi-volume routine at initialization time.

Disk input. Records are read from disk to the main storage input area until G is full. As the records are read, a check is made for an end-of-volume condition. After a record is read, the disk address of the next input record is built. This address may be on the same track or on the next track and/or cylinder.

Tape input. Records are read from tape and checked for end-of-volume and full G area. Error records are bypassed, if so specified by the user.

Fixed-length records with user exits, variable-length records, or records requiring data conversion have their addresses built one at a time as they are processed.

INMOVE, BD-B1

Registers are loaded with start and end addresses of the input area and unsorted records are moved to the start of the input area. This function is also used to move variable-length records to the output area.

Note: This routine is not entered the first time through phase 1.

P1INBG, BD-E1

Note: This description is for disk input. For tape input, the tape-input routine (Chart BE) will have been relocated to P1INBG.

When the end of a cylinder is reached, the address of the next cylinder is constructed (at DINJCY). If end-of-volume is reached, a branch is made to the checkpoint sub-routine, DINMVA. If neither condition exists, CCW's are built (at DIHERE) for reading the count fields of the G records. The routine then continues to DIBLRD.

DINMVA, BD-D4

Phase 1 mainline is written onto the checkpoint track and the multi-volume routine (DSORT101) is read into main storage. After the multi-volume routine has been executed, it is written back onto the checkpoint track and the phase 1 mainline is read into main storage.

DIBLRD, BD-A2

The read CCW is initialized with information supplied by the read count chain and records are read from the disk input area. If end-of-file is detected, a branch is made to the multi-volume routine via DINMVA. When G area is full, a branch is made to INEXIT. If end-of-cylinder is detected, or if G area is not full, the routine returns to P1INBG.

INFINI, BD-H5

This function is entered when the final volume of the final file has been read; it is used to calculate the final G. The number of doublets is determined and saved in NRDUB, and the end-of-input switch (IBVNE4) in the output routine is set so that the next overlay can be fetched at that time.

If the output will consist of more than one block, a padding switch (PADSW) is set so that the last output block will be filled with F's (for ascending records) or with 0's (for descending records). A branch is made to INEXIT for fixed-length records, or to ONEXIT for variable-length records.

INEXIT, BD-J5

The current end-of-input-area address is saved in INMAX and the record count is updated and saved in RDCNT. A branch is then made to PILEV1 in the internal sort routine (Chart BF).

ONEXIT, BD-J4

This sub-routine is entered for variable-length records and, if required, for user exit 12 and data conversion; initialization was done at VLRINZ (Chart BC). Addresses are built for all records in G area and a branch is made to PILEV1 (Chart BF).

TAPE INPUT ROUTINE - BE

TINBG, BE-B3

For tape input, this routine is relocated at P1INBG.

Tape records are read in and tests are made for end-of-volume, checkpoint records, errors, and full G area. Checkpoint records are bypassed and the next record is read. If end-of-volume is detected, a branch is made to the multi-volume routine via DINMVA. If an error is detected, the record is bypassed, if specified by the user (exit 13), and the bypassed-record count is incremented and stored in INTBYP. If the bypass option is not specified, operator intervention is allowed.

When the G area is full, a branch is made to INEXIT (Chart BD). At end-of-file(s) time, the bypassed-record count is printed and a branch is made to INFINI (Chart BD).

INTERNAL SORT - BF

Records in the G area are sorted and merged by the internal-sort routine, which is divided into levels 1 and 2.

Note: The records are not physically sorted; their addresses are placed in sequence in address tables. In the following illustrations, records are shown in the address tables for clarification only.

Level 1. Addresses are built for two fixed-length records located in the G area. (The addresses for variable-length records were built at ONEXIT, Chart BD.) The control fields of the two records are compared; the address of the winning record is placed in one of two address tables, followed by the address of the losing record. The two-address sequence in the address table is referred to as a "doublet" (Figure 20). The sequence that determines the winning and losing record is designated by the SORT control card. The address-building and sorting processes continue until the G records have been sorted into doublets.

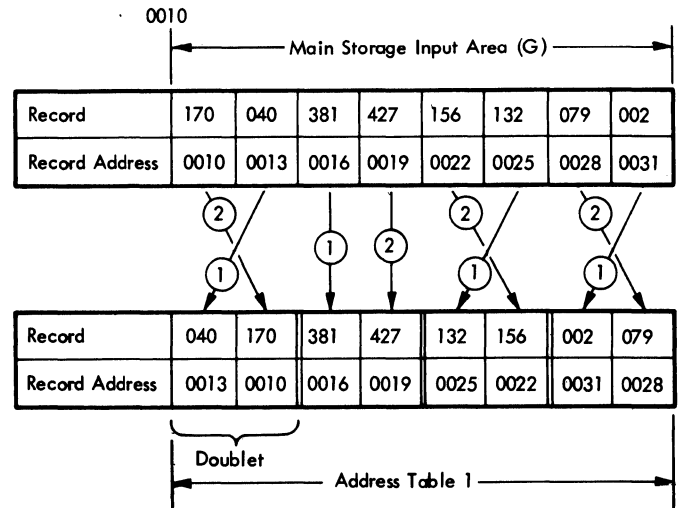


Figure 20. Doublets

Level 2. Registers are initialized for the merge operation. The control fields of the records represented by the first address of each of the first two sequences (doublets) are compared. The address of the winning record is stored in the second address table. Comparison is then made between the record represented by the second address in the winning sequence and that represented by the first address of the losing sequence. Again, the address of the winning record is stored in the second address table (Figure 21). Merging continues until the end of one of the sequences is reached, at which time the addresses remaining in the alternate sequence are copied into the second address table. The process is repeated for the next two sequences and continues until all sequences from the first address table have been transferred to the second address table. (A sequence may be called a "string" when it contains more than two records.)

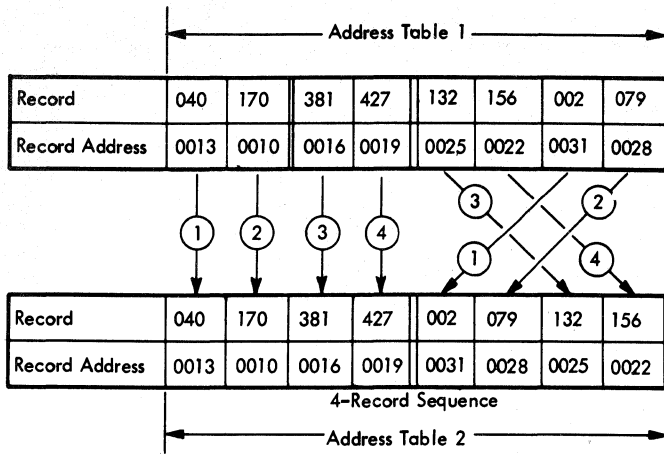


Figure 21. 4-Record Sequences

When all sequences have been transferred, the address tables are reversed, and the 4-record sequences are merged into 8-record sequences (Figure 22).

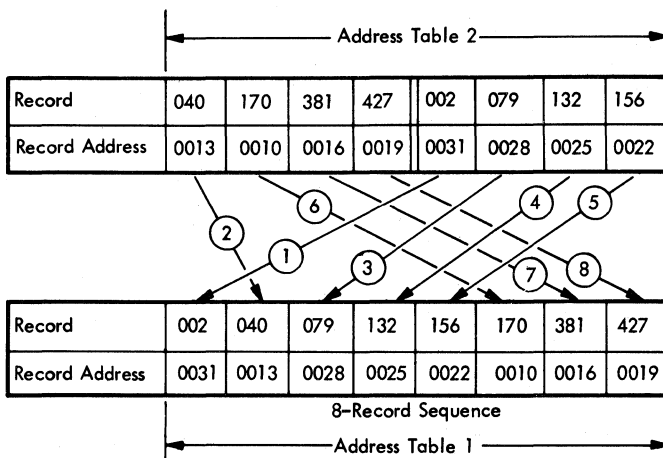


Figure 22. 8-Record Sequences

When only two strings remain to be merged in an address table, control is passed to the output routine (Chart BG). The records represented by the addresses in the remaining strings are then merged into the output area (Figure 23).

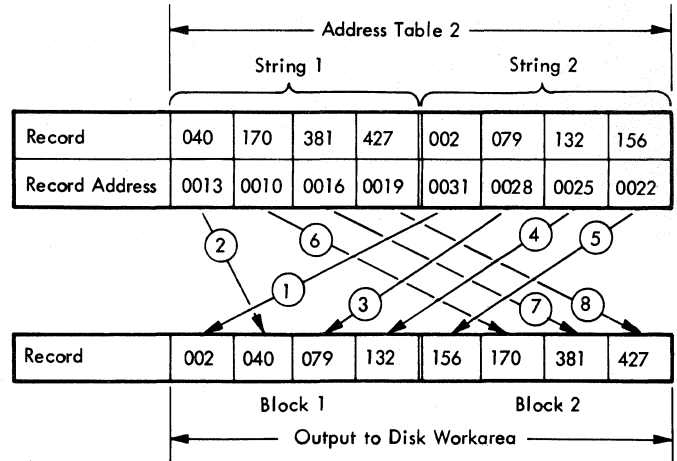


Figure 23. Output

P1LEV1, BF-B2

Addresses are built for G records (fixed-length) and doublets are created for processing in level 2. For variable-length records, addresses for all G records have been built at ONEXIT (Chart BD).

A flag is entered at the end of the last doublet created from G and the routine continues to IBVIF4.

IBVIF4, BF-C2

The level 2 registers are loaded with branch addresses. A branch is made to IBVJH2 to get the address of next doublet from the address table and the routine continues at CMPCHN.

CMPCHN, BF-D2

Control fields of two records are compared; when the winning record is determined, a branch is made to STORM1 or STORM2, as the case may be. When two strings or less remain, the branches will be to STO1ZZ or STO2ZZ and the program continues to the output routine (Chart BG).

STORM1-STORM2, BF-E3

If the record from the doublet of string 1 is the winner, STORM1 is entered to store that address in the second address table.

The string base register is incremented by 4 and if neither string is depleted, a branch is made to CMPCHN.

Entry to STORM2 is made when the record from string 2 is the winner. The processing is the same as that for STORM1.

If string 1 is depleted, but not string 2, a branch is made to STORM2; if string 2 is depleted, but not string 1, the branch is to STORM1. When both strings have run out, a branch is made to EXIT.

EXIT, BF-G3

The base registers for string 1 and string 2 are initialized. When two strings are left in the input address table, the condition is indicated by a '03' in string 2. A branch is made to CMPCHN for compare, then to IBVJC2.

REDOUT, BF-H5

If only one string remains in the input address table, the routine initializes to move that remaining string to the output area. A branch is then made to IBVJC2.

IBVJC2, BF-J4

The input/output address tables are reversed and a test is made to determine if two or less strings remain in one of the address tables. If there are more than two strings remaining a branch is made to CMPCHN; if not, the last pass branches (STO1ZZ, STO2ZZ) are initialized before branching to CMPCHN.

OUTPUT ROUTINE - BG

The output routine merges the remaining two sequences into the output area instead of into an address table. When the output area is full, it is written in the disk work area. The DSHIFT table is used for storing the starting disk address of each sequence in a set. (A set is the number of sequences, equal to OM, order of merge, that the next phase will use in a merge.) The address for block 1 of a sequence is moved from the DSHIFT table to a storage area, where the addresses of succeeding blocks are built. As the blocks per track or cylinder number in an address is exceeded, the complement of the record number or cylinder number is added (Figure 24). If the current disk address just built exceeds the upper work limit, a new work area extent is brought in from the WKTAB table, and a new current disk address is computed.

After all blocks of a G are written on disk, the current disk address is saved in the lower section of the DSHIFT table. All other sections of this table move up, with the uppermost section having the starting disk address for the next sequence of the set. Control is passed to the input routine for the reading of the next G records.

It is possible that the size of the input file is such that the final output block cannot be filled. When this is the case, a padding routine fills in the final output block so that it can be written in the disk work area. Ascending control fields pad a block with hex F's (1-bits) and descending control fields pad with zeros so that these will be considered "losing" records. A record count is kept so that padding records may be dropped in phase 3.

4 = Order of Merge
5 = Blocks Per Track

Max of 60 Sections

DSHIFT Table

		C	H	H	R	C	H	H	R
Displacement		8	0	7	1	A	0	0	0
		8	0	7	2	A	0	0	0
		8	0	7	3	A	0	0	0
		8	0	7	4	A	0	0	0

-Before-

PUB Table Lower Limit Upper Limit

Max of 60 Sections

		C	H	H	R	C	H	H	R
Displacement		8	0	7	2	A	0	0	0
		8	0	7	3	A	0	0	0
		8	0	7	4	A	0	0	0
		9	0	0	2	A	0	0	0

-After-

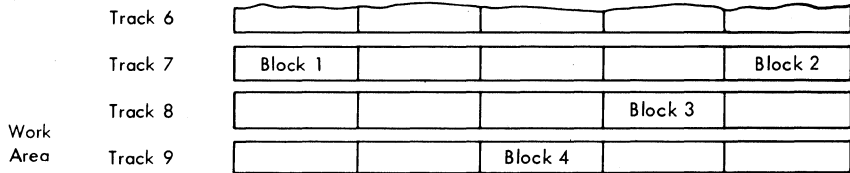
- 1 Starting Address
Add Order of Merge
- 2 Is Record Number Valid
(Equal to or Less Than BPT)
- 3 Add Order of Merge
- 4 Record Number Is Invalid
Add Complement of BPT
to Address (05=FB)
- 5 Is Record Number Valid
- 6 Add Order of Merge
- 7 Record Number Is Invalid
Add Complement of BPT
to Address (05=FB)
- 8 Is Record Number Valid
- 9 Add Order of Merge
- 10 Record Number Is Invalid
Add Complement of BPT
to Address (05=FB)
- 11 Track Number Invalid
Add Complement of Track
- 12 Address of 1st Sequence
of Next Set

Current Disk Address

C	H	H	R
08	00	07	01
08	00	07	05
			+4
08	00	07	09
00	00	00	FB
08	00	08	04
			+4
08	00	08	08
00	00	00	FB
08	00	09	03
			+4
08	00	09	07
00	00	00	FB
08	00	0A	02
00	FF	F6	00
09	00	00	02

← Block 1
← Block 2
← Block 3
← Block 4 (G)
Records Processed

Cylinder 8



Cylinder 9

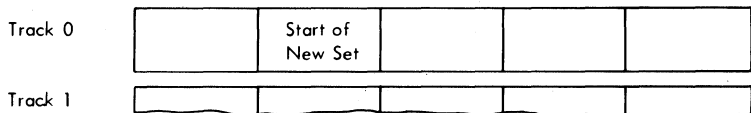
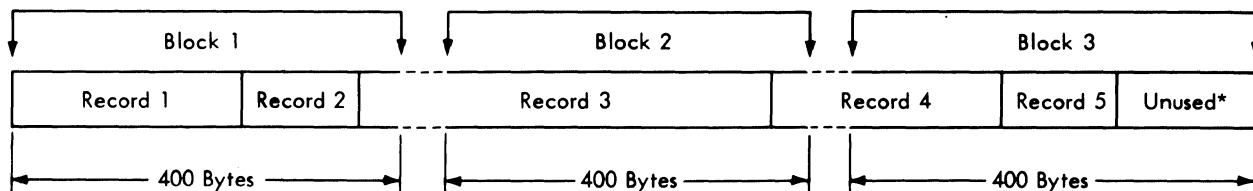


Figure 24. Address Creation

Example: 3-400 Character Sort Blocks in a Sequence
 6-Variable-Length Input Records
 Record 1 - 250 Bytes
 Record 2 - 100 Bytes
 Record 3 - 400 Bytes
 Record 4 - 200 Bytes
 Record 5 - 100 Bytes
 Record 6 - 200 Bytes



* Record 6 is not outputted because it will not fit in the last block of the sequence.

Figure 25. Output, Variable-Length Records

The output routine for variable length differs in that it may be necessary to split records between blocks (Figure 25).

When the end of the file is reached, overlay 5 (DSORT105) is fetched.

MOPSI, BG-B2

Records represented by the winning addresses are moved into the output sort block and a branch is made to IBVMC3.

IBVMC3, BG-C2

As long as the output area is not full, a branch is made to IBVME2 to restore the level 2 registers and to return to the location stored in XLINK (STO1ZZ+8 or STO2ZZ+8).

When the output area is full, G count is reduced by sort block count and tests are made for end of string and for padding records. If padding is not required, a branch is made to IBVMH4. Padding records are used for fixed-length records when the last sort block is not completely filled. The padding routine is entered by a branch to PADRTN.

PADRTN, BG-E3

Records are built as losing records, depending on the sequence specified in the SORT control card. Zeros are padded for descending sequence; 1-bits (hex F) for ascending sequence. The records are moved to the output area until it is full. The routine then continues to IBVMH4.

IBVMH4, BG-F2

An EXCP macro is issued and the output block is written on disk. The next disk address is created using BPT (blocks per track) and the work area limits. Calculations are checked for validity of tracks and cylinder. (For a more detailed description of address creation, see Figure 24 and accompanying text.) The routine then continues to IBVPB2.

IBVPB2, BG-H2

If the block count is equal to the order of merge times the number of blocks per string, it indicates the end of a set; a branch is made to IBV001 where the current disk address is saved (in OUTRD) as the starting address of the new set. A branch is then made to IBVNE4. If it is not the end of a set, the current block count is saved and the routine continues to IBVVLRL.

For an end-of-string condition, a branch is made to IBVND4 where the DSHIFT table is initialized to use the next address for the start of the next sequence. The DSHIFT table is shifted so that the next starting address is located at the start of the table. The routine then continues to IBVNE4.

If not the end of a string, a branch is made to IBVMF2 to restore registers and to return to the location in XLINK (STO1ZZ or STOR2ZZ).

IBVNE4, BG-J4

A branch is made to INMOVE in the input routine (Chart BF) until the end-of-file is reached. Then the routine initializes for the next overlay and branches to FETCH where DSORT105 is fetched.

When the end-of-file is reached, phase 1 executes the compression routine and updates the checkpoint record.

The compression routine checks the number of blocks in the last set of phase 1. Gaps between blocks exist if the number of blocks do not equal order of merge times number of blocks per sequence (Figure 26). A table labeled IBVTAB stores the number of blocks for each sequence in the last set.

For example, in Figure 26, the first sequence has three blocks, the second sequence has three blocks, the third sequence has two blocks, and the fourth sequence has no blocks. The count is stored in IBVTAB table. The compression is accomplished through the use of read and write commands. A read command is given starting with the first block address of the last set. IBVTAB indicates that there are three blocks in sequence 1. The write

- 4 -- Order of Merge
- 3 -- Blocks Per Sequence
- 8 -- Blocks in Last Set

Table IBVTAB

0	3	0	3	0	2	0	0	FF	00
Sequence W		Sequence X		Sequence Y		Sequence Z			

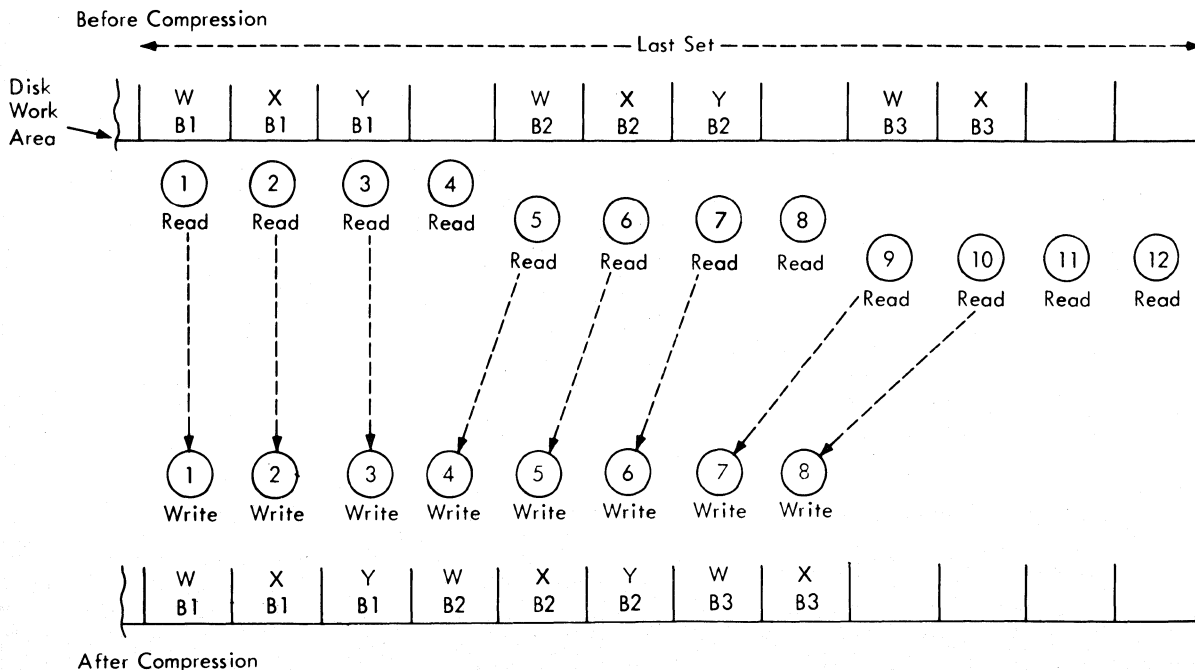


Figure 26. Phase 1 Compression

command is also initialized with the address of the first block in the last set. Because both addresses are the same, no transfer of data is needed. The read and write commands are updated to the next block in the set. IBVTAB indicates that there are three blocks in sequence 2. Read command number 2 is given. Because there is a block written here and the write address is the same, no data is transferred.

This process continues until the fourth sequence is reached. Table IBVTAB indicates that no blocks are written on disk for sequence 4. The read command address is updated to the next block, while the write command address stays the same. As a read command is given for a block, the corresponding area in IBVTAB is decremented by one. Reading and writing continues until all sections in IBVTAB are zeros, indicating the end of compression of blocks in the last set.

The final routine executed by phase 1 computes constants and updates the checkpoint record. The constants computed for phase 2 are:

- Number of passes.
- Number of sequences.
- Number of blocks in last set.
- Number of blocks in last sequence.
- Number of records processed.

The program then fetches the specified overlay of phase 2.

IBV024, BH-B2

Overlay 5 is relocated to the end of the mainline (the end of the output routine).

CMPRES, BH-C2

If compression is required, the table IBVTAB is initialized with the number of blocks in each sequence of the last set. The program continues in the compression routine until the records in the last set are compressed. For a more detailed description of the compression routine, see Figure 26 and accompanying text.

The routine then continues at EXITP1.

EXITP1, BH-D2

Constants are calculated for use in the next phases:

- Number of passes (NRPAS).
- Number of sequences (NRSEQ).
- Number of blocks in last set (PB2BKT).
- Number of blocks in last sequence (NRBLS).
- Number of records processed (RDCNT).

The checkpoint record is read in, updated with the constants just calculated, and written back on the checkpoint track.

OV2INW, BH-E2

An EXCP macro is issued and the end-of-phase messages are printed. The overlay number at the FETCH macro is initialized for the specified record type and order of merge.

FETCH, BH-F2

The specified overlay of phase 2 is fetched.

INITIALIZATION FOR ADDRROUT RUN - BJ

The initialization routine for an ADDRROUT-type sort run is similar to the initialization routine for tape or disk input.

The CHNMVE routine is initialized with the number of control fields, their lengths, and their displacement within the record.

The checkpoint record is updated with:

- Input record length.
- Number of control data fields.
- Length of control fields.
- Location of control fields.

IBV023, BJ-B2

The checkpoint record is read in and placed at the end of phase 1, overlay 3.

VLRINZ, BJ-C2

If data conversion and/or exit 12 is specified, the routine is relocated. If user exit 12 is specified, these functions are bypassed by a branch to TAGINT.

TAGINT, BJ-E2

The move code in the CHNMVE routine is initialized with the number of control fields, their lengths, and their displacement within the record.

TAGCHK, BJ-F2

The checkpoint record is updated with:

- Input record length -- Total length of all control fields plus 10 for address.
- Control data fields -- Number of control fields. (If the control fields are in binary format, it is possible to reduce their number. For example, if there are three CF's to begin with and all are in ascending sequence, the number of control fields for the sort equals 1. If the first and third are ascending and the second is descending, the number of control fields equals three.)
- Length -- The number of bytes in one or more successive control fields with the same collating sequence, if in binary format.
- Location -- The location of a control field is its relative position within the tag. (A tag is designated as the 10-byte address followed by the control fields.) The first CF is located in the eleventh byte.

The type of input is determined and the input routine is initialized for fixed- or variable-length records and key.

IBVAA1, BJ-A4

The internal sort routine is initialized. The values for G, number of sort blocks in G, and record-length (computed by the assignment phase) are relocated for use by the internal-sort routine. The number of doublets is computed and saved in NRDUB.

CHNCMP, BJ-B4

Constants supplied by the assignment phase are used to initialize the compare chain with the number of control fields and the displacement and length of each field. The compare string is relocated.

If conversion is specified, the conversion routine is relocated and initialized with the number of control fields and the location of the compare chain. For fixed-length input, the move routine is initialized with the record-length.

LOCARE, BJ-C4

Using G supplied by assignment phase, calculations are made to determine the areas necessary for the address tables, the start and end addresses of the input area, and the start address of the output area. The compare chain is initialized using the number, length, and displacement of control fields.

OUTIZ, BJ-D4

The output routine is initialized for interleaving. The disk address calculation routine is initialized to compute disk addresses for the sort run. The CCW and CCB are initialized, and the starting addresses for the first order of merge are calculated and placed in the DSHIFT table. A branch is then made to LBLCHK (Chart BL).

INPUT ROUTINE FOR ADDRROUT RUN - BK, BL

The input routine is initialized and a block of records is read from disk. After a record is read into the input area, a tag is built consisting of the record disk address and the control field(s). When the tag or RAF area is full, control is passed to the internal-sort routine (Chart BF).

VARBLK, BK-B2

A test is made to determine if the input volume limit is exceeded and, if so, the end-of-volume switch is turned on and a branch is made to DINMVA. If the limit is not exceeded, the count of the first record is read (at RDCNT) to determine the length of the block. The routine then continues to P1INBG.

P1INBG, BK-B1

The ADDRROUT input routine is initialized with disk input record address, main storage input area address, and RAF starting address.

DINGET, BK-C1

A block of records is read from disk and a test is made for end-of-volume. If end-of-file, a branch is made to the checkpoint routine at LBLCHK (Chart BL). If not end-of-volume, the routine continues to DINGUD.

DINGUD, BK-E1

The next input block is located on disk. For variable blocking, if end of cylinder is detected, a branch is made to VARBLK. For fixed-length or variable-length records when end of cylinder is not detected, the routine continues to DINLES.

DINLES, BK-G1

If end-of-extent condition is detected, a branch is made to DINMVA; if not, the routine continues to BLDADR.

BLDADR, BK-H1

The next record address is built and tested to determine if the maximum block size has been exceeded. If exceeded, a branch is made to the WLR routine to print WLR message, bypass the record, and return to P1INBG. If not exceeded, a branch is made to TAGMNL.

DINMVA, BK-H2

Registers are restored and the end-of-file switch is turned on. A branch is then made to TAGMNL.

TAGMNL, BK-K2

The sort mainline is initialized with the calculated G, the start address of the input area, and the start address of the RAF (record address file).

TAGADR, BK-A4

The 10-byte disk address portion of the tag (MBBCCHRDD) is prepared:

M = Pack number (0-244)
BB = Bin number (always 00)
CC = Cylinder number (0-199)
HH = Head number (0-9)
R = Number of blocks per track (1-50)
DD = Zero for unblocked fixed-length records, or the displacement (in bytes) of the record within the block.

CHNMVE, BK-B4

The designated control field(s) are moved from the input record to the RAF area, following the disk address just built.

NXTREC, BK-C4

The record length is stored and the block size is incremented by the record length. Then, if the end of the input block has not been reached, a branch is made to NOEOB. If the end of the input block has been reached but end-of-file has not been reached, a branch is made to LBLCHK; if end-of-file has been reached, a test is made for a full RAF area. If the RAF area is full, a branch is made to INEXIT (Chart BL); if not full, the branch is to TAGADR.

NOEOB, BK-E5

The record address is incremented by the record length and the next address for tag is calculated. A test is then made for a full RAF area; the resulting branches are the same as in the preceding block, NXTREC.

LBLCHK, BL-B3

Phase 1 mainline is written onto the checkpoint track and the multi-volume label routine (DSORT101) is read into main storage. After the multi-volume routine has been executed, it is written back onto the checkpoint track and the phase 1 mainline is read back into main storage.

INFINI, BL-F2

This function is entered when the final volume has been read; it is used to calculate the final G. The number of doublets is determined and saved in NRDUB, and the end-of-volume switch (IBVNE4) in the output routine is set so that the next overlay can be fetched at that time.

If the output will consist of more than one block, a padding switch (PADSW) is set so that the last block will be filled with F's (for ascending records) or with 0's (for descending records).

ONEXIT, BL-H3

This sub-routine is entered for user exit 12 and for data conversion, when required; initialization was done at VLRINZ (Chart BJ). Addresses are built for all records in the G area and a branch is made to P1LEV1 (Chart BF).

INEXIT, BL-H1

The record count is updated and saved in RCDCNT. A branch is then made to P1LEV1 (Chart BF).

INTERNAL SORT (ADDROUT RUN) - BF

The internal-sort routine for ADDROUT run is the same as that for disk or tape input. For details, refer to Chart BF and its narrative.

OUTPUT ROUTINE (ADDROUT RUN) - BG

The output routine for ADDROUT run is the same as that for disk or tape input. For details, refer to Chart BG and its narrative.

MULTI-VOLUME, EXIT 11 LINKAGE - BM, BN

The multi-volume routine opens, closes, or processes end-of-volume (EOV) routines for disk or tape input files and ADDROUT input. The file type may be:

- Disk - standard labels without user labels.
- Tape - standard labels without user labels, non-standard labels, or unlabeled.

The routine, initialized according to user specifications, is called in from the checkpoint track to process the specific condition. After execution, the routine is written back on the checkpoint track and the mainline is read back into main storage.

MNLDSK, BM-B3

For tape input, a branch is made to MNLTAP. For disk input, a test is made for end-of-file (EOF). If EOF, the selected bit is set in the disk DTF table and the routine continues at OPENA.

OPENA, BM-D3

When end-of-file condition is not indicated by the mainline, control is passed to the IOCS - OPEN routine. The next file to be processed is opened, standard and user labels are checked, and the end-of-file condition is tested. For EOF, a branch is made to ENDINP; otherwise, the routine continues at GETXT.

GETXT, BM-G3

The log unit address and the upper and lower limits of the input area in main storage (extents) are saved. Control is then returned to the phase 1 mainline at DINMVA for regular sort or at LBLCHK for ADDRROUT sort.

ENDINP, BM-G2

A test is made to determine if the EOF was for the last input file. If so, the last-file flag and last-file switch are set and the routine continues to DCLOSE.

DCLOSE, BM-H2

The current input file is closed. If it was not the last input file, the next file name is initialized in the Disk DTF and a branch is made to OPEN. If the last-file switch (LSTFLSW) is on, control is returned to the phase 1 mainline at DINMVA for regular sort or at LBLCHK for ADDRROUT sort.

MNLTAP, BM-C4

For the initial open, a branch is made to OPENT. For all subsequent entries for open, the branch is to EOVEOF (Chart BN).

OPENT, BM-D4

The log unit address and file name are updated and placed in the tape DTF table. The tape is then opened by IOCS and control is returned at MNLLINK.

MNLLINK, BM-H4

The log unit address is saved and control is returned to the phase 1 mainline at DINMVA (Chart BD).

EOVEOF, BN-B1

The block count is placed in the tape DTF

table and a test is made for end-of-file (last volume). If not end-of-file, the volume number is saved and a branch is made to FEOVBR. For end-of-file, the volume table is shifted, the number of volumes in the next file is obtained, and the EOF switch (byte number 32) is turned on in the tape DTF. The routine then continues to FEOVBR.

FEOVBR, BN-F1

For end-of-volume/end-of-file, or end-of-volume and not end-of-file with standard labels, a branch is made to ENDMAC. For end-of-volume/not end-of-file and not standard labels, the branch is to FEOVRTN.

FEOVRTN, BN-J1

This routine uses the FEOV macro to force an end-of-volume condition. FEOV causes automatic volume switching. The log unit address is placed in a register and control is returned to the phase 1 mainline at DINMVA (Chart BD).

ENDMAC, BN-G2

Trailer labels are checked and, if end-of-file, a branch is made to EOFADDR; if not, the log unit address is placed in a register and control is returned to the phase 1 mainline at DINMVA (Chart BD).

EOFADDR, BN-G3

If the EOF is for the last input file, the last-file indicator and last-file switch are set and the routine continues to NXTFILE.

NXTFILE, BN-H3

Control is passed to the IOCS-CLOSE routine to close the input file. If the last-file switch (LSTFILSW) is not on, a branch is made to OPENT (Chart BM). If the last-file switch is on, control is returned to the phase 1 mainline at DINMVA (Chart BD).

PHASE 2 - EXTERNAL SORT OR MERGE

Phase 2 consists of one of four overlays, depending on the order of merge and the input record type:

- DSORT201 (4-way merge, fixed-length), for an order of merge from 1 to 4 and either a fixed-length record sort or the ADDROUT option for fixed- or variable-length records.
- DSORT202 (7-way merge, fixed-length), for an order of merge from 5 to 7 and either fixed-length record sort or the ADDROUT option for fixed- or variable-length records.
- DSORT203 (3-way merge, variable-length), for an order of merge from 1 to 3 and variable-length records.
- DSORT204 (6-way merge, variable-length), for an order of merge from 4 to 6 and variable-length records.

The overlay to be used in phase 2 is determined and called in by phase 1. Phase 2 then calls in the corresponding overlay for phase 3.

For the purpose of describing the program logic, this phase has been divided into two general categories:

- Fixed-length records (7-way and 4-way merges), Charts CA through CM
- Variable-length records (6-way and 3-way merges), Charts CN through CY

This introduction serves for both categories. Where necessary, duplicate figures and charts are provided (with the required differences, if any) so that each category is complete in itself. For example, there are two major-component charts (03) and two main storage layout figures.

The sequences created in phase 1 are merged in phase 2. The order of merge (M) represents the number of input sequences that will be merged into one output sequence during each merge. The order of merge, which was determined by the assignment phase, is considered to be the fastest possible sort for the input file.

Strings or sequences are read from the input portion of the work area, merged together, then written in the output portion of the work area. When all strings from the input portion of the work area have been merged into the output portion of the work area, a pass is complete. For the next pass, the input half of the work area becomes the output half and vice-versa.

At the start of phase 2, pass 1, the sequences developed in phase 1 reside in the input portion of the disk work area. A test is made comparing the number of sequences (at the start of each pass) against the order of merge. If the number of sequences is equal to or less than the order of merge, the upcoming pass is the last one and phase 3 is fetched from the core image library. If the upcoming pass is not the last one, merging begins in phase 2.

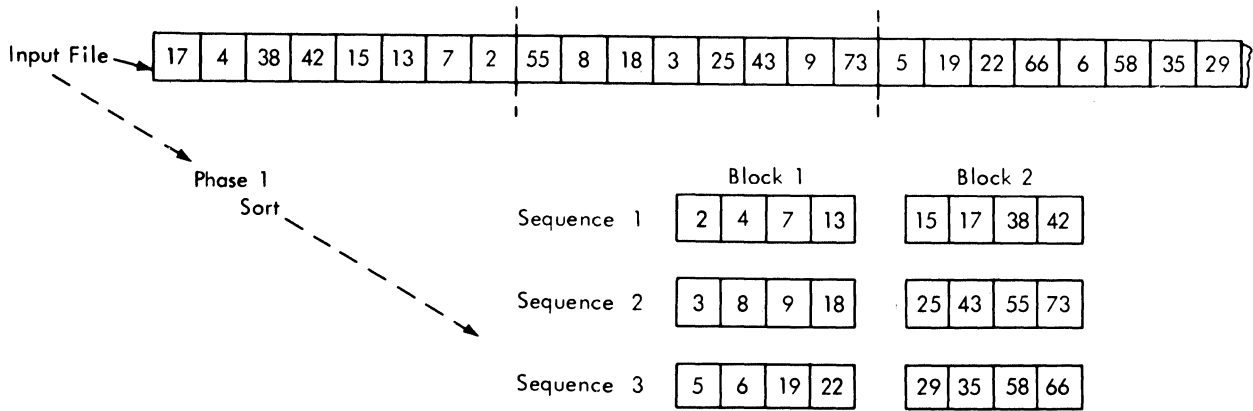


Figure 27. Sort Blocks

As stated in phase 1, sequences are placed in the disk work area in a blocked format. Block 1 is the lowest alphameric block of each sequence in an ascending order (Figure 27).

The size of each input area depends on the type of record being processed. For fixed-length records, each input area is

equal to the length of a block from the input portion of the disk work area, as illustrated in the merge example in Figure 28. For variable-length records, each input area has an additional overflow area, immediately preceding it (see Figure 29, item 1). The overflow area is equal in size to the maximum logical record length minus one (LMAX-1).

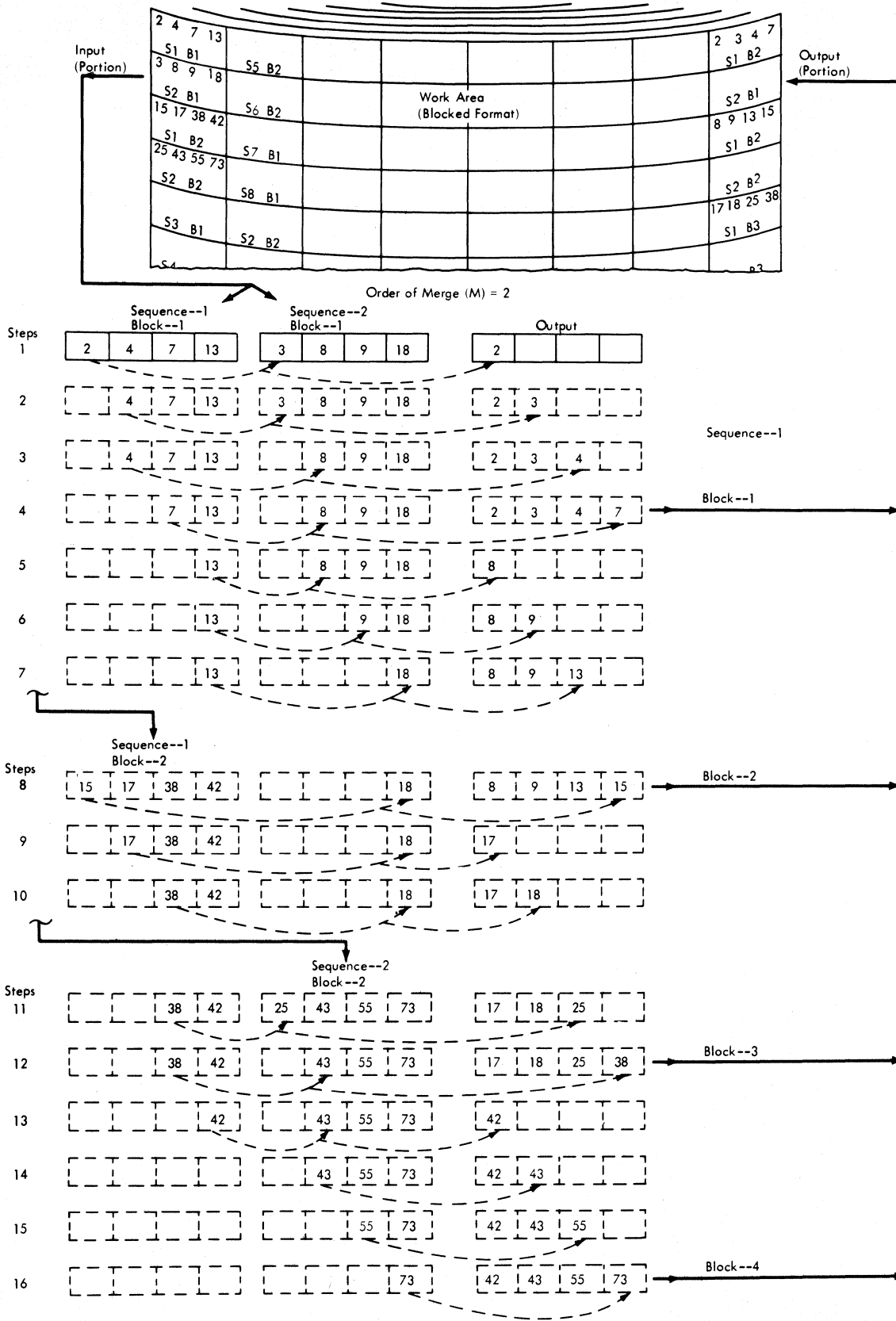


Figure 28. Phase 2 Merge

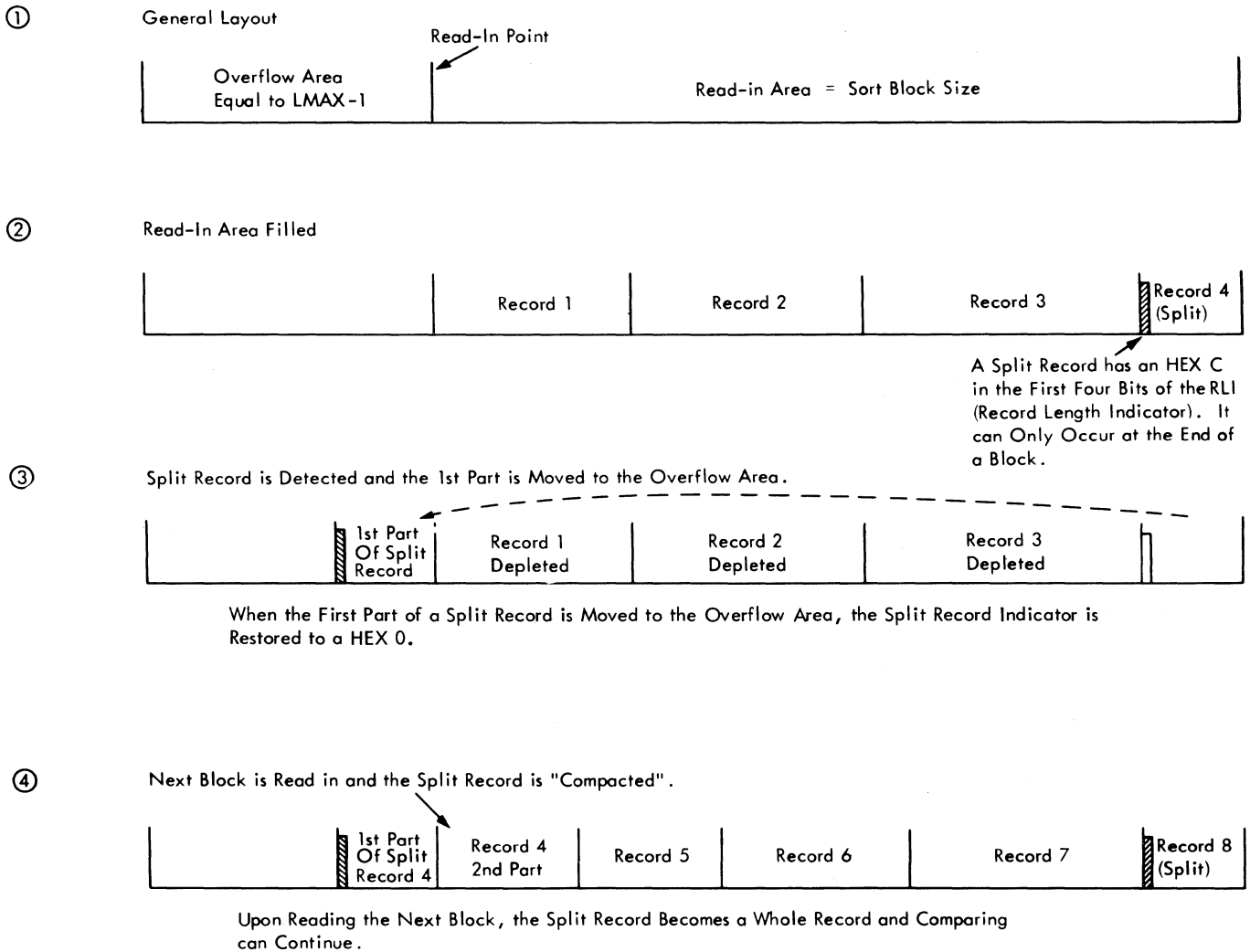


Figure 29. Variable-Length Record Input Area

Because variable-length records are read and written in fixed-length blocks, phase 2 (as well as phase 3) must process split records in the overflow areas. A split record is one which is divided into two portions with one portion in each of two blocks. Before such a record can be compared for merging, it must be rejoined or compacted.

Phase 1, in outputting sequences, creates a split record (identified by a hexadecimal C in the first four bits of the RLI) whenever a winning record cannot wholly fit into an output block. As many bytes as can fit into the unfilled portion of the block are moved. The record is then marked as a split record, the block is written, and the remaining portion is then moved to the start of the next block.

In phase 2, each time a winning record is moved to the output area the next record

in that input block is tested for a split condition. If it is a split record, the remaining bytes within the block (a split record can be encountered only as the last record in a block) are moved to the overflow area; the split condition is erased and the next block of the sequence is read into main storage, compacting the split record. The merging process can then continue (see Figure 29, items 2, 3, and 4).

An example of a 2-way merge is illustrated in Figure 28. The first blocks (B1) from each of the first M sequences (M = 2) are read from the input portion of the disk work area into the main storage input areas.

Merging of the two sequences will begin with the first record of S1B1 (sequence 1 block 1) being compared against the first record in S2B1. Working on an ascending

order, the lowest record is moved into the output area (Figure 30, step 1).

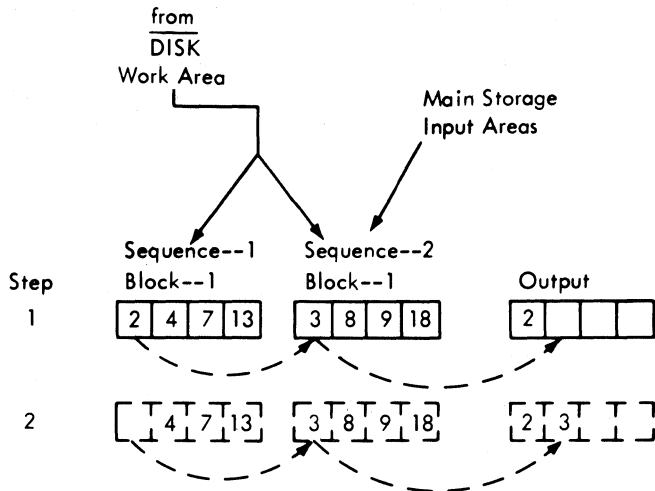


Figure 30. Main Storage Output Area

The second record in S1B1 is now compared against the first record in S2B1 (Figure 28, step 2). Again, the lowest is moved to the output area. Merging of records in the input areas continues until the output area is full, or until one of the input areas is depleted.

When the output area is full, it is written in the output disk work area as a block of this new sequence.

Figure 28, step 7, shows S1B1 (sequence 1, block 1) being depleted as a result of a compare operation. The next sequential block of the sequence using that area is read in. Merging continues with the first record of S1B2 being compared against the appropriate record of S2B1 (Figure 28, step 8).

This cycle continues until all blocks of the M sequences now using the main storage input areas have been merged into one sequence. This is called a merge within a pass.

At this time, the next M sequences are read in and merged as described. This is repeated until all sequences have been read from the input disk work area into main storage input areas, merged, and written in the output portion of the disk work area. This is the end of a pass.

A checkpoint record is updated at the start of each phase 2 pass.

When an end-of-pass condition exists but the upcoming pass is not the last pass, the phase 2 mainline is reinitialized for a new pass. Pointers to the input and output work areas are reversed so that the output strings of the previous pass become the input strings for the upcoming pass.

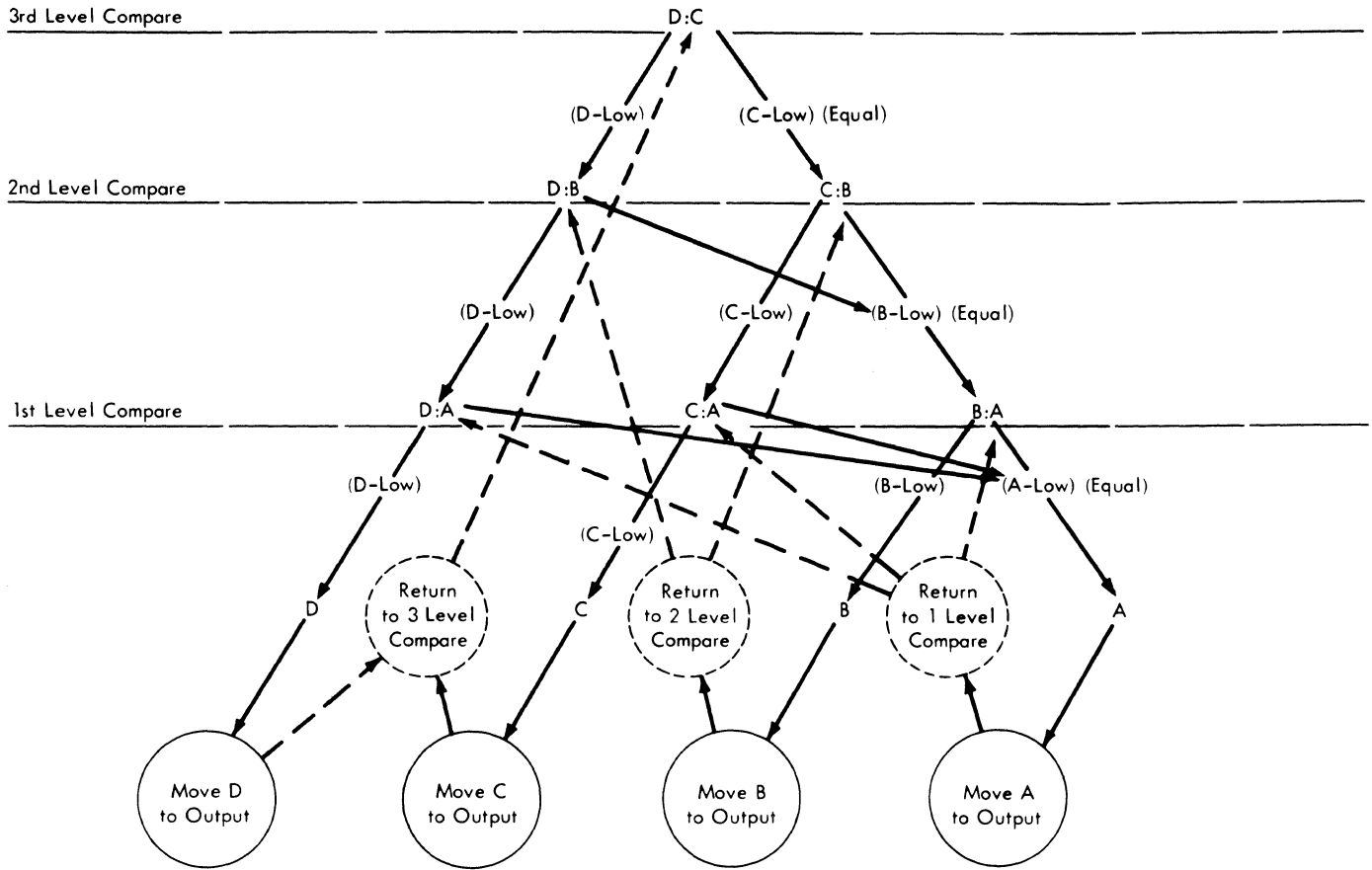
If the end-of-pass condition exists and the upcoming pass is the last pass, phase 3 is fetched into main storage.

The "compare tree" in Figure 31 illustrates the compare operation used for merging.

With M blocks in the input areas, the first record of each block is compared. When the winning record is found it is moved to the output area. Figure 31 shows four input areas, D, C, B, and A, filled with two records for each block. Using the compare tree and the input area example 1, record 1 of sequence D is compared against the first record of the other sequences and found to be the winning record. Because D was the winning record, the next compare must start again with D:C.

In example two of Figure 31, the sequence C record is found to be the winner. The next compare must start again with D:C. In example three of Figure 31, sequence B is the winning record; in this case, because sequence C record was compared to sequence B record, C must be lower than record D. Therefore, the next compare is started at level 2, C:B. In example four, sequence record A is the winning record. Comparing is returned to level 1 (B:A) because sequence B record was already found to be lower than either sequence D or C record.

The interleaved output technique that is used in this program is illustrated in Figure 32, using a 3-way merge as an example. In phase 1, the input file was sorted into a total of 38 sequences. The gap factor or interleave factor is, for the most part, equal to the order of merge. It may, however, be reduced in the later stages of a pass.



Example	Sequence	D	C	B	A	Output Record
1		8 9	10 13	12 16	14 20	D-- (8)
2		8 9	7 10	12 16	14 20	C-- (7)
3		8 9	7 10	6 16	14 20	B-- (6)
4		8 9	7 10	6 16	5 20	A-- (5)

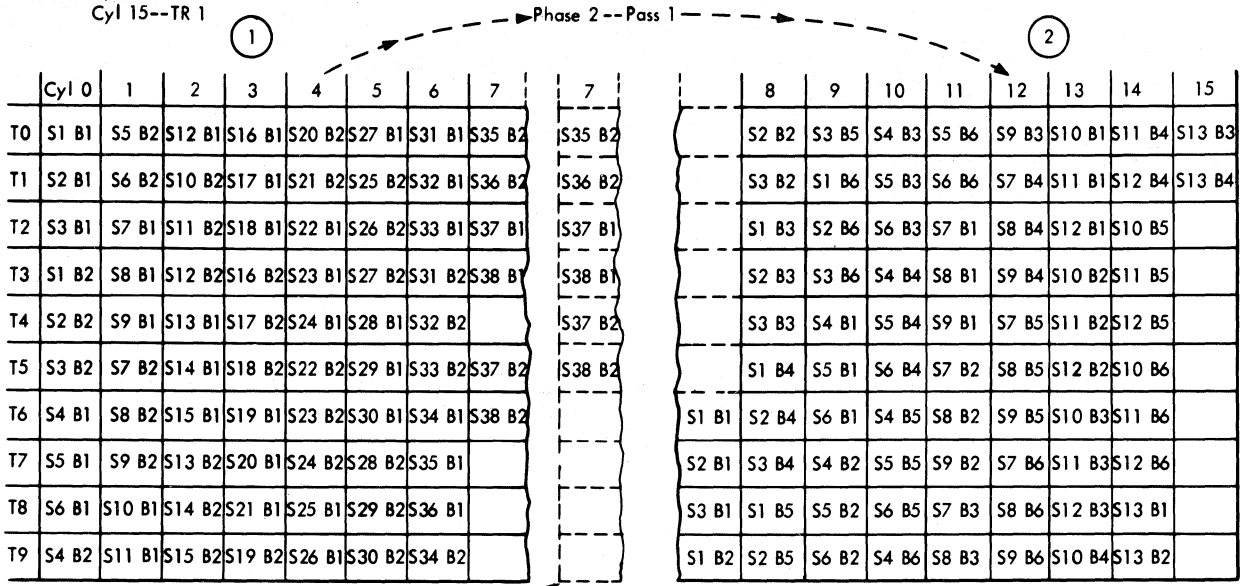
If D or C is moved to the output area, 3 compares are required before another record can be moved.

If B is moved to the output area, 2 compares are required before another record can be moved.

If A is moved to the output area, 1 compare is required before another record can be moved.

Figure 31. Compare Tree

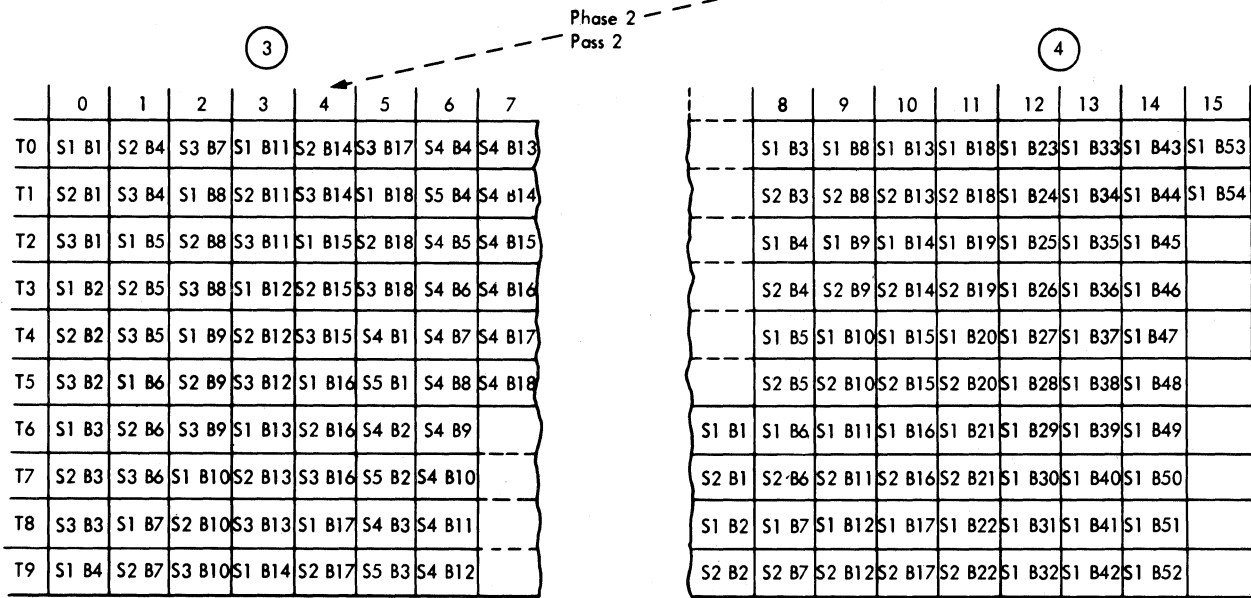
Order of Merge=3
 Total Work Area=Cyl 0--TR 0
 Cyl 15--TR 1



Output of Phase 1 (Input to Pass 2)

Cylinder 7 After
Pass 1 Compression

(Pass 1 Output is
Pass 2 Input)



(Pass 2 Output is
Pass 3 Input)

(Pass 3 Output is
Phase 3 Input)

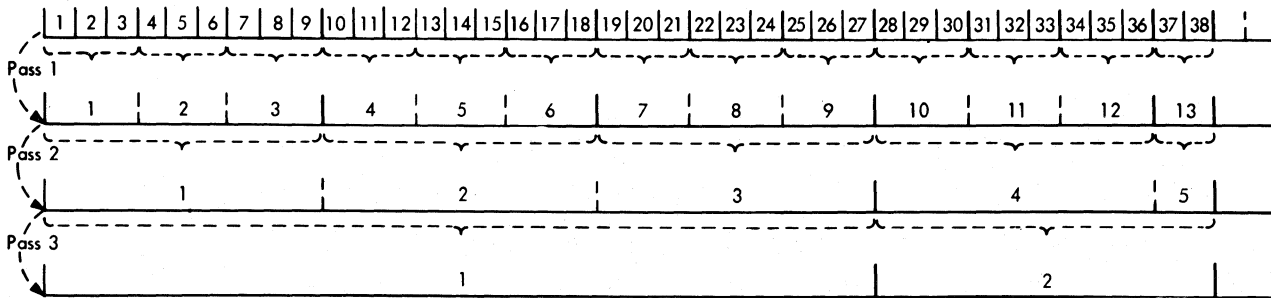


Figure 32. Interleaving (3-Way Merge)

Note that a gap exists between S38B1 and S37B2 (Figure 32, section 1). Because the actual file size was not known in phase 1, S37 and S38 were interleaved for a 3-way merge. After S38B2 had been written and the end-of-file detected, phase 1 compressed S37 and S38 so that the gap would no longer exist. Sequences 37 and 38 comprise the last merge of phase 2, pass 1, and have an interleave factor of 2. The information concerning the last merge is passed on to phase 2 within the checkpoint record. Phase 2 merges the output of phase 1 using the specified order of merge. Figure 32 shows that during the first pass, 38 sequences are merged into 13 sequences. The interleave factor (F) for the first 12 merges is equal to M; in the last merge, it is equal to two. The interleave factor is used for input and output and can vary individually.

The factors which cause a reduction in (F) are:

- Input
 1. Number of sequences which will comprise the last merge of a pass. The last merge of a pass is initiated whenever the number of sequences remaining is equal to or less than the phase 2 order of merge, i.e., when $S \leq M$.
- Output
 1. Number of sequences remaining on input after each set of M^2 input sequences have been merged into M output sequences. If, after merging M^2 input sequences, the number of remaining sequences is equal to or less than M^2 , a reduction in output interleaving may have to be implemented.
 2. The number of blocks which comprise the last merge of a pass; that is, the total number of blocks that will be merged into an output sequence during the last merge of a pass.

Figure 32, section 1, is the input to phase 2, pass 1. As the last merge of pass 1 is detected, the input interleave factor is reduced from three to two for sequences 37 and 38. The input interleave factor is two, while the output interleave factor is one. Pass 1 determines how many input

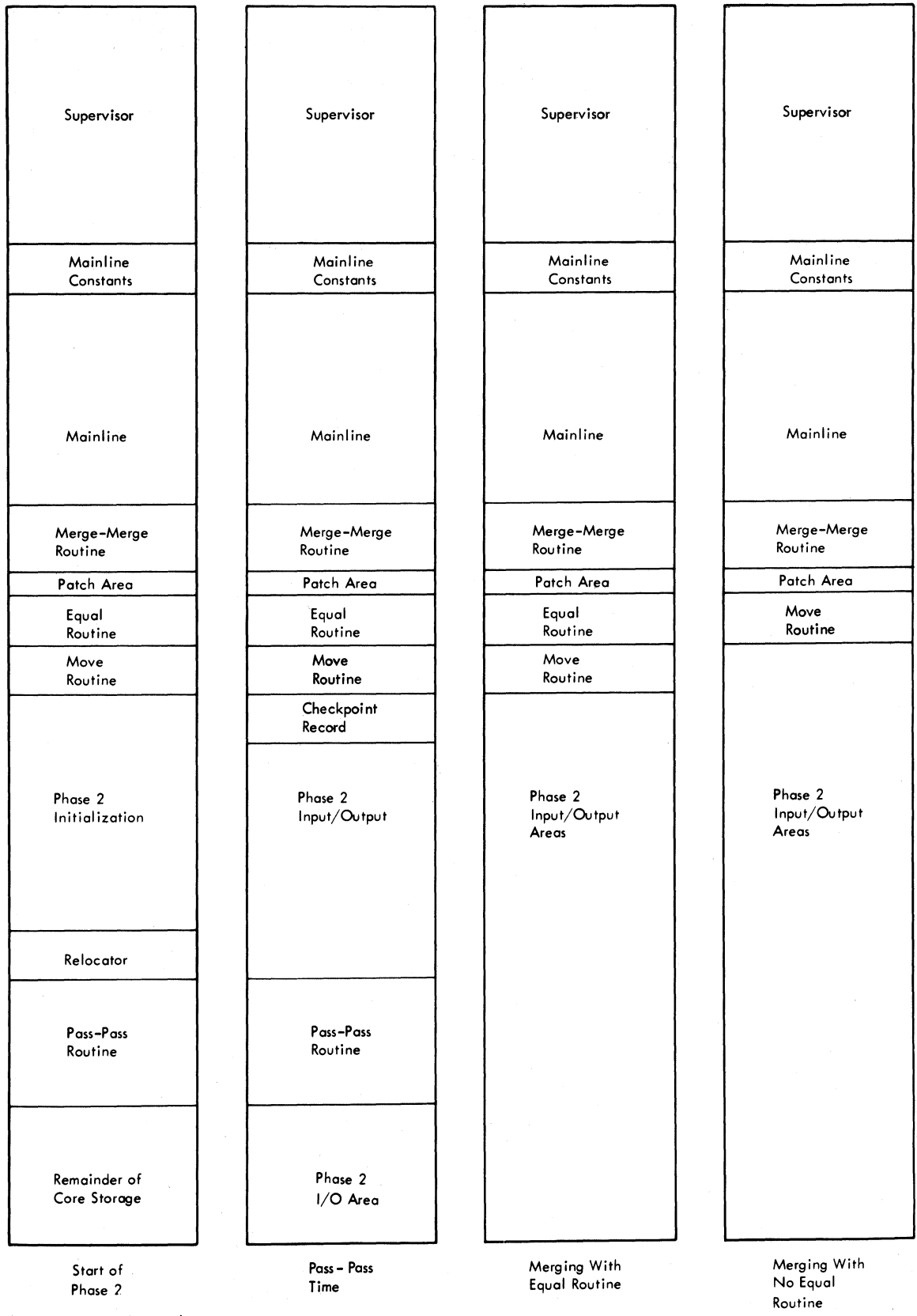
sequences remain before attempting to merge M^2 sequences. In other words, pass 1 checked when there were 38, 29, 20, 11, and 2 sequences remaining on input. The first time that M^2 or less sequences were detected was when there were two sequences remaining. Output interleave factor is reduced to one, while the input factor is reduced to two. Pass 1 alerts pass 2 that there are four blocks in the last merge of pass 2, and that there will be a 1-way merge on input.

Section 2 of Figure 32 is the output of pass 1, or the input to pass 2. The output interleave factor changes when the remaining sequences are M^2 or less, or when the remaining sequences equal four in this example. Output (F) is reduced to two. Note that output sequence 4 will contain 18 blocks whereas sequence 5 will contain only four blocks. Output interleave factor for sequence 4 will be two only until the block count is one more than the total number of blocks in the last sequence. It will then be reduced to one for the remainder of sequence 4. For sequence 5, the output interleave factor goes back to two. Input interleave factor is reduced to one for the last merge, per information left by pass 1. Pass 2 output contains five sequences for two merge runs. Sequence and block count information of the last merge is passed to pass 3.

Pass 3 starts with less than M^2 sequences. Therefore, the output interleave factor starts at less than three (in this case, two). Pass 2 information contains the number of blocks for the last merge of pass 3. When the first sequence block count exceeds the total block count of the last merge, (sequences 4-5) the output factor for the first sequence reduces to one (Figure 32, section 4: S1B24). The output interleave factor for the last merge is two; the input factor starts at two, then is reduced to one.

The next pass (4) is the final merge, performed by phase 3. The input interleave factor starts at two, then is reduced to one after the twenty-third input block of sequence 1. Output is written consecutively on the user-specified unit, tape or disk.

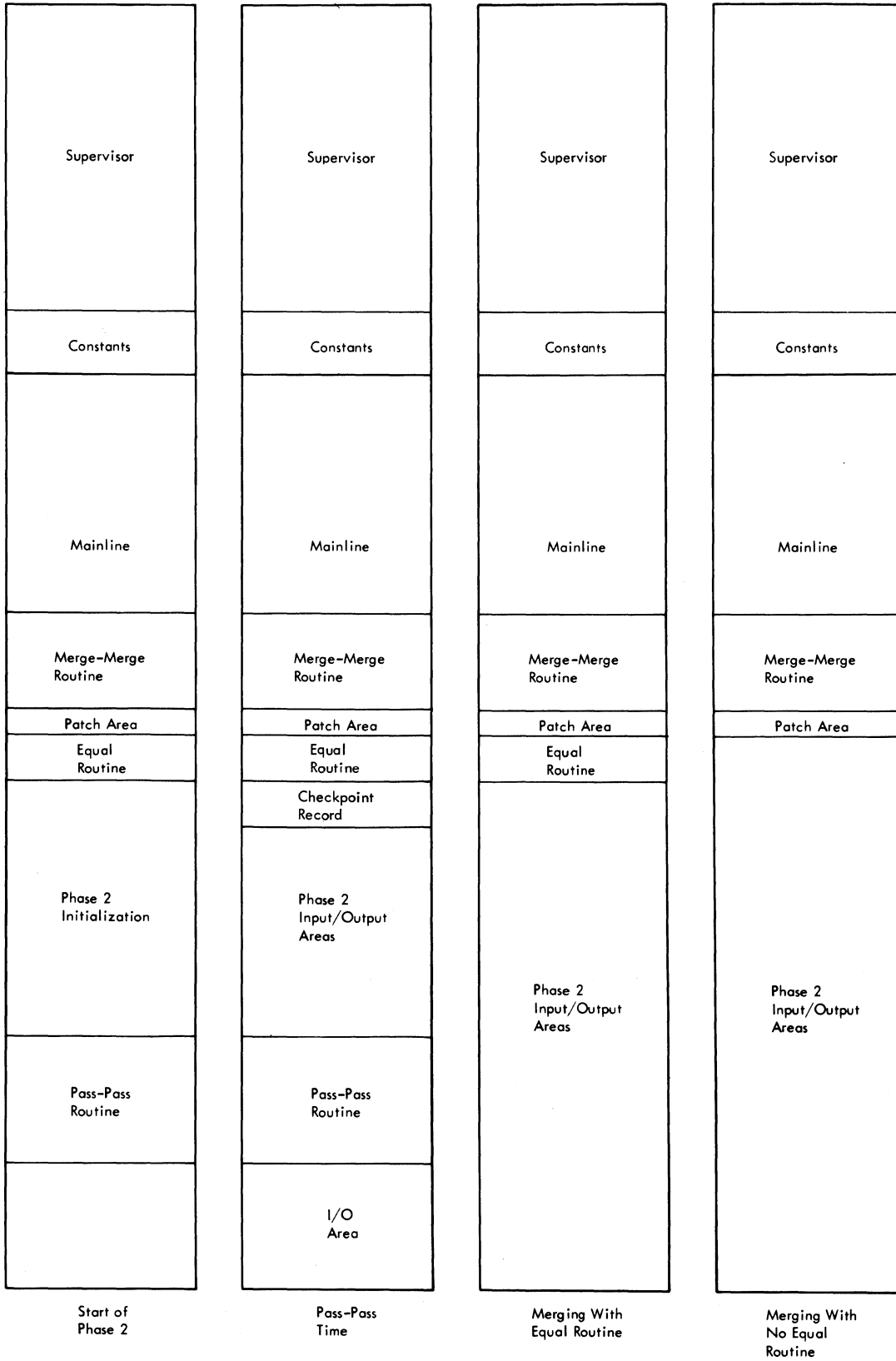
Figures 33 and 34 illustrate phase 2 main storage layouts for fixed- and variable-length records, respectively.



Note: Not drawn to scale.

Figure 33. Phase 2 Main Storage Layout for Fixed-Length Records

Move Routine
Incorporated
in Mainline



Note: Not drawn to scale.

Figure 34. Phase 2 Main Storage for Variable-Length Records

PHASE 2 INITIALIZATION, FIXED-LENGTH RECORDS - CA

The checkpoint record, created by the assignment phase and updated by phase 1, is read into main storage to obtain information to be used in phase 2. This routine then initializes:

- Merge-merge routine
- Pass-pass routine
- Sequence compare loops
- Interleave address routine
- Relocatable routines
- Constants

Merge-merge. Initialized according to the order of merge that was calculated by the assignment phase (from 2 to 7).

Pass-pass. Initialized according to the order of merge to be used during phase 2. Input and output interleave factors, as well as pointers to the logical halves of the work area table, are stored at this time.

Compare. All compares within the compare loops are initialized with the length and location of the first control data field. Branches are initialized for either ascending or descending sequence.

Interleave address routine. Factors are calculated for the input/output disk address interleaving for phase 2.

Relocatable routines. The equal routine is not relocated. However, if the equal routine is not required or if the records contain less than 12 control fields, the move routine is relocated by the required amount (the amount that the equal routine is shortened). When control is returned by the relocater, initialization continues and the pass-pass routine is written out on the 2311 checkpoint track as record 2. Subsequently, it is brought into main storage only when an end-of-pass condition exists.

The next step is to calculate or allocate the input/output areas to be used in the merging process. The number of main storage areas needed is equal to the order of merge plus one. Each area is equal to the sort-block size. The input/output channel programs are then initialized with the number of bytes to be transferred during an I/O operation (number of bytes is equal to the sort-block size). The program

then continues to the merge-merge routine (Chart CB).

INTPH2, CA-B2

The base register (register 11) is loaded with the starting address of phase 2. This address is obtained from CKPCCB+8. Registers 2 and 3, which were loaded at the end of phase 1, are stored:

<u>Req.</u>	<u>Contents</u>	<u>Stored in</u>
2	Logical unit address of device where checkpoint is written.	1. CCB for taking checkpoint at every pass (RSTCCB). 2. CCB for reading checkpoint (CKPCCB). 3. CCB for reading and writing the pass-pass routine (PPCCB).
3	Disk address of checkpoint	1. Location CHECKP, for updating checkpoint record and for passing this address to phase 3. 2. Location CHHR, current disk interleave address. Used only to read the checkpoint record for initialization.

RCHKPT, CA-C2

An EXCP macro is issued and the checkpoint record is read into location PASSNO. When the read operation is completed, the routine continues to RDCPOK.

RDCPOK, CA-D2

A test is made to determine if the equal routine is required (more than one control field per record). If so, a branch is made to COMPIT; if not, the branches in the mainline compare loops are initialized to

bypass the branch-and-link to the equal routine when two equal records are encountered.

COMPIT, CA-F2

The length and displacement (location within record) of the first control field are OR'ed into the compare instructions in the mainline compare loops. The instructions that determine the branching conditions in the compare loops are then initialized for ascending or descending sequence.

The mainline is then initialized for the record length (from SORTL), which is equal to either the input record length (L1) specified in the RECORD control statement or, if the ADDRROUT option is specified, to ten plus the total length of all control fields (CF + 10).

The order of merge (M) and blocks per track (BPT1) are obtained from the checkpoint record and stored in constants PH2IOM and BPT, respectively. Registers are then initialized for calculating interleave factors for disk addresses.

DIVAGN, CA-H2

The interleave factors are calculated by dividing the order of merge by the sort blocks per track. The quotient is stored in location NQUOT and the remainder in location NRMDR. The order of merge is then reduced by one and the next factors are calculated and stored adjacent to the first. This process is repeated for a number of times that is equal to the order of merge. These factors will be used by the interleaved disk address routine; the quotients for calculating the head or track numbers, the remainders for calculating the record numbers.

The merge-merge routine is initialized with a switch (at WAY4) for a specified order of merge for phase 2.

The pass-pass routine is then initialized with indexing factors (from POINTI) that point to the logical halves of the work area table.

A test is made to determine if a copy pass is required in phase 2. If the user has specified that the output file is to be written in the first half of the disk work area, and if an inaccurate file size has been specified on the SORT control

statement, the input for phase 3 could be in the first half of the work area (based on the number of passes calculated by phase 1). When this condition exists, an extra pass must be made so that the phase 3 input will be in the second half of the work area.

OUPTOK, CA-D3

The physical limits of the work area are obtained from the checkpoint record (LOGPHY) and placed in the LIMITS table. The move routine and the equal routine indicator bits are set in RLCOND and a branch is made to the relocater routine (Chart FA). After the equal routine is included (if required) and its length is determined, the move routine is relocated, if necessary, and control is returned to this routine at START.

START, CA-F3

The pass-pass routine is initialized with the starting address of the phase 2 input and output areas in main storage. This address, which will later be the address into which the checkpoint record is read, is obtained from location RLISA after the relocater routine is executed.

If the sort blocks per track (BPT) is equal to 1, the second half of the work area was not formatted by phase 1. When such is the case, instructions are now modified so that pass 1 of phase 2 will format the second half of the disk work area, which is the pass 1 output portion.

The pass-pass routine is then written on the checkpoint track as record 2.

CPIOAS, CA-B4

Constants for phase 2 input and output areas are now calculated. These are:

- ABEGIN through GBEGIN - Starting addresses of input areas A, B, C, D, E, F, and G (as required by order of merge).
- AEND through GEND - Ending address of input areas A, B, C, D, E, F, and G (as required by order of merge).
- OBEGIN - Starting address of output area.

This instruction is a no-op except when the BPT is greater than 1, in which case a branch is made to NOWCKD. When BPT = 1 and the format must be established for the output portion of the disk work area, the routine allocates the output count field next to the output area and increases the output block count by eight.

The ending address of the output area (OUTEND) is calculated and stored. The read and write CCW's are then initialized with the input and output data counts and a branch is made to the merge-merge routine (Chart CB).

MERGE-MERGE ROUTINE, FIXED-LENGTH RECORDS - CB

The merge-merge routine initializes the mainline to merge the next set of input sequences into one output sequence. A set is equal in number to the OM (order of merge) calculated by the assignment phase, except during the last merge of a pass when a set may have less sequences than the OM.

The functions performed by this routine vary with the conditions at time of entry. These conditions are:

- End-of-pass, which occurs (1) at the start of phase 2 and (2) each time all the input sequences have been merged into output sequences. For example, if phase 1 output is 64 sequences and the OM is 4, and end-of-pass condition would exist (1) at the start of phase 2, (2) when 64 input sequences have been merged into 16 output sequences, and (3) when the 16 new input sequences (the output from the previous pass) have been merged into 4 new output sequences.
- End-of-merge with more than M^2 input sequences remaining.
- End-of-merge with M^2 or less input sequences remaining.
- End-of-merge with M or less input sequences remaining (which signifies that the upcoming merge is the last one of the current pass).

The functions performed for each condition are:

End-of-pass. The last-merge switches in the input and output routines are turned off and a branch is made to the pass-pass routine. When control is returned to this routine, there would no longer be an end-of-pass condition; at this point, it would be either (1) end-of-merge with more than M^2 sequences or (2) end-of-merge with M^2 or less sequences. These functions are described in the following paragraphs under their respective headings. Note that the only function that will not be performed in an end-of-pass condition is the shifting of the output disk address table. This table is never shifted before the first merge of a pass.

End-of-merge with more than M^2 sequences. The mainline is re-initialized for the order of merge calculated by the assignment phase and the number of input sequences is reduced by a number equal to the order of merge. The number of merges to be done before new output interleave factors are implemented is reduced by one. Then the output disk address table is shifted to obtain the starting address for the next output sequence.

End-of-merge with M^2 or less sequences. If this is the last merge of a pass, the functions performed are described in the next paragraph. If not the last merge, the mainline is initialized with new interleave factors to be used in computing the disk output addresses for the remainder of the pass. The output sequences created from this point on will become the input sequences for the last merge of the next pass.

End-of-merge with M or less sequences. When this condition is reached, it is the last merge of a pass. The input interleave factors are changed and the end-of-pass switch is turned on. The program will then continue to the pass-pass routine at the completion of the last merge.

The end-of-merge switch (MMPP1 in the interleave factors routine, Chart CM) is turned off to remain off until next time an end-of-merge condition is detected. Turning off switch MMPP1 consists of activating the branch to the move routine.

MMPPS, CB-C3

If this routine is being entered on an end-of-pass condition, a branch is made to LM1234. For an end-of-merge condition, the end-of-sequence indicator (hexadecimal EF) is cleared from the end of the output area and registers are initialized to open the mainline and fill the input areas (registers SAVEA through SAVED, depending on OM).

The input sequence counter (SR) is tested to determine if the next merge will be the last one of the pass ($SR \leq OM$). If not, a branch is made to WAY4; if so, the routine continues to REDUCEI.

REDUCEI, CB-C2

Instructions at IRMDR and IQUOT are initialized with new input interleave factors for the last merge. The end-of-pass switch at MMPPS is turned on (it will be turned off for the next pass in the pass-pass routine) and the routine continues to LM1234.

LM1234, CB-D3

One of two courses of action is taken at this point:

- Start of last merge - Turn on last merge input interleave switches (LM1 through LM_n, depending on OM) and continue to WAY4-4.
- End of pass - Turn off switches LM1 through LM_n and branch to LMOSW.

At WAY4-4, the pass-pass routine has inserted an instruction that initializes the mainline with the order of merge that is to be used during the last merge of a pass.

WAY4, CB-E4

The input sequence counter (SR) is decremented by a factor equal to the order of merge. Location MAXFAC is then tested to determine if the output interleave factors are to be changed (see Appendix A for description of MAXFAC contents). When the count in MAXFAC is at zero, it signifies that M^2 or less input sequences remain to be merged. Under these

conditions, new output interleave factors are to be implemented and the routine continues to REDUCEO. As long as the count in MAXFAC is not zero, the count is decremented by one and the routine branches to NOCHG.

REDUCEO, CB-E2

Instructions at ORMDR and OQUOT are initialized with new output interleave factors, which have been pre-determined by the pass-pass routine. Continue to LMOSW.

LMOSW, CB-F3

At the end of a pass, the output interleave switch is turned off (LMO in the output routine, Chart CL). This switch had been turned on (branch 00) during the pass when the number of input sequences remaining was equal to or less than M^2 (when the count in MAXFAC was reduced to zero). After switch LMO is turned off, a branch is made to ENDPAS if it is the end of a pass; otherwise, the routine continues to NOCHG.

ENDPAS, CB-H3

At the end of each pass, as well as at the start of phase 2 (which is considered an end-of-pass condition), the pass-pass routine is read into the main storage I/O areas by channel program PPCHPG. The program then branches to EXECPP in the pass-pass routine, executes the routine, and returns control to this point in the merge-merge routine. The channel program then writes the pass-pass routine back on the checkpoint track in the work area and a branch is made to MMPPS+4 to start merging in the new pass.

NOCHG, CB-H4

For each new merge within a pass, the count for MAXFAC (which was just decremented at REDUCEO-4) is stored back in MAXFAC. Except for the first merge of every pass, the output disk address table (ORADDR) is shifted to give the starting disk address for the new output sequence. The output sequence block counter (LMSTRG) is re-initialized with a count equal to the number of blocks contained in the sequences which comprise the last merge of the pass.

This counter is used at LMO in the output routine to detect when the output interleave factors should be reduced. The program then branches to USTOPA to open the mainline compare routine for sequence A.

PASS-PASS ROUTINE, FIXED-LENGTH RECORDS - CC

At the beginning of each phase-2 pass, the pass-pass routine is read from the checkpoint track into the main storage input/output areas at location EXECPP. The routine first reads the checkpoint record, updates it, and restores it to the checkpoint track. It then computes the initial disk addresses for M sequences for both input and output. The two pointers to the logical halves of the work area table are reversed so that initial disk addresses can be computed at the beginning of the next pass. Next, the interleave factors are restored to their original values, which were computed during phase 2 initialization (order of merge divided by BPT). The number of the pass being entered is listed on SYSLOG. If the upcoming pass is the last one, a switch is set to enable job control to fetch phase 3 into main storage.

The pass-pass routine then calculates the interleave factors to be used (1) during the last merge of the pass for input and (2) during the last set of merges for output (when M^2 or less input sequences are merged into M or less output sequences). The merge-merge routine is initialized with a switch that determines the order of merge during the last merge of the pass.

At the end of the pass-pass routine, control is returned to the merge-merge routine which writes the pass-pass routine back on the checkpoint track (record 2).

EXECPP, CC-B2

The pass-pass routine is entered each time the entire file has been passed through or merged into a new set of sequences. It initializes phase 2 for the upcoming pass (except the last).

The end-of-pass switch (MMPPS in the merge-merge routine), is turned off (branch 00) and the routine initializes to read the checkpoint record into storage.

TAKECP, CC-C2

The checkpoint record is read into the first 316 bytes of the phase 2 input/output areas. This address was obtained from RLISA during the initialization routine.

CKPTOK, CC-D2

The checkpoint record is updated with:

1. A decimal integer representing the phase 2 pass number (PH2PAS)
2. The number of passes remaining (NOPASS)
3. The number of sequences to be merged during the upcoming pass (NSR)
4. The number of sort blocks that comprise the last merge of the pass (LMBLOK)
5. The number of sort blocks contained in the last sequence of the pass (MARGEL)
6. Two hexadecimal pointers that reflect the logical halves of the work area (POINTL).

The updated checkpoint record is then written back on the checkpoint track.

CPOADR-8, CC-F2

The initial disk addresses for the input and output sequences of the upcoming pass are computed for a maximum order of merge (four or seven, as the case may be). The addresses are stored in ARADDR-GRADDR for input, and starting at ORADDR for output. As each initial address is calculated, two values are extracted from the work area table:

1. An index value (multiple of 12), or pointer to the work area table (LIMITS).
2. The logical unit address pertaining to the address.

These values are placed adjacent to the disk address in the table.

SHTPTR, CC-G2

The two hexadecimal pointers to the logical halves of the work area table are reversed. The input area for the current or upcoming pass becomes the output area for the next pass, etc., alternating between the two halves throughout the phase.

NEWITL, CC-H2

The interleave factors to be used during the pass are calculated ($M/BPT = Q + R$) and stored at IRMDR+1 and IQUOT+1 for input and at ORMDR+1 and OQUOT+1 for output. These factors are used until it is determined later in the pass that they are to be changed.

NEXTPASS, CC-J2

If the upcoming pass is the last one, the switch at NOTLAS is made a no-op to enable the pass-pass routine to fetch phase 3 into main storage. Until such time, NOTLAS remains a branch to ENDPAS in the merge-merge routine (Chart CB). The number of the upcoming pass (whether for phase 2 or phase 3) is then printed out:

'7DB1I PHASE 2, PASS nn'
or '7DC1I PHASE 3, PASS nn'

ILEAVE-10, CC-B3

The number of output sequences to be formed in the upcoming pass is calculated (NSR).

Note: The output sequences in this pass will be the input sequences for the next pass.

INITOM, CC-C3

The input interleave factors that will be used in the last merge of the pass (when $S \leq OM$) are now calculated and stored in REDUCEI+1 and REDUCEI+5. These are two instructions at label REDUCEI in the merge-merge routine that will place these input interleave factors in IRMDR+1 and IQUOT+1, respectively, when the last merge is entered. The reduced input interleave factors to be used when the order of merge

is reduced are stored in IRMDR1+1 and IQUOT1+1.

The merge-merge routine is then initialized with the order of merge to be used during the last merge of the pass. This consists of inserting one of seven possible instructions (which are listed beginning at LASTM) at WAY4-4 in the merge-merge routine.

OLEAVE, CC-E3

The number of merges to be performed during the next pass before the output interleave factors are to be changed is now calculated and stored in MAXFAC. The output interleave factors to be used when MAXFAC is reduced to zero (when $S \leq M^2$) are stored in REDUCEO+1 and REDUCEO+5. These are two instructions at label REDUCEO in the merge-merge routine that will place these output interleave factors in ORMDR+1 and OQUOT+1, respectively, when MAXFAC=0. The reduced output interleave factors to be used when the order of merge is reduced are stored in ORMDR1+1 and OQUOT1+1.

NOTLAS, CC-F3

As noted in the text under label NEXTPASS, this location is a branch to ENDPAS in the merge-merge routine until it is determined that the upcoming pass will be the last pass. NOTLAS is then made a no-op and the routine assembles and writes the constants for phase 3 on the checkpoint track and fetches phase 3 into main storage.

INPUT ROUTINE, FIXED-LENGTH RECORDS - CD

This routine fills the input areas in main storage with records from the input portion of the disk work area. At the beginning of a merge, all the input areas are filled and the compare loops are initialized. Subsequently, the input areas are refilled individually as they are depleted and, as input sequences are depleted, the compare loops are closed off one by one.

The maximum number of input sequences (A through G for a 7-way merge or A through D for a 4-way merge) are called in. As each block is read into its input area, the disk address of the next block in that sequence is calculated by a separate routine (Chart CM). The interleave factors for calculating the disk addresses are changed when necessary.

The number of sequences that are called into the input areas is determined by the order of merge. When all the main storage input areas are filled, the program continues to the compare loops (Charts CE through CK, as the case may be).

USTOPA, CD-B2

The compare loops are initialized for the required sequences and a corresponding bit is set in the end-of-merge indicator (OMERGE):

Seg.	Label	Initialize Branches at	OMERGE Bit
A	USTOPA	BPUTG, BPUTF, BPUTE, BPUTD, BPUTC, BPUTB, FILLA	1
B	USTOPB	COMPBA, BGA, BFA, BEA, BDA, BCA, FILLB	2
C	USTOPC	COMPCB, BGB, BFB, BEB, BDB, FILLC	3
D	USTOPD	COMPDC, BGC, BFC, BEC, FILLD	4
E	USTOPE	COMPED, BGD, BFD, FILLE	5
F	USTOPF	COMPFE, BGE, FILLF	6
G	USTOPG	COMPGF, FILLG	7

When all the specified sequences have been processed and the 1-bits in OMERGE have been inverted to 0-bits as described in FILLA, a branch is made to the output routine (Chart CL). Until then, the routine continues to FILLA (or FILLB, etc., as the case may be).

FILLA, CD-D2

The unconditional branch at this location is a no-op as long as there are records to be processed in the current input sequence; the routine thus continues to GETA (or GETB, etc. as the case may be). When the end of the current input sequence is reached, the compare loop for that sequence is closed off and the corresponding bit in OMERGE is inverted to a zero. Then, as long as there are more input sequences, a branch is made to an address that varies

according to the current sequence and the order of merge. These addresses are listed in the table on Chart CD.

GETA, CD-E2

This location is entered not only from the preceding function block (FILLA) but also from the various compare loops as long as there are records in the input sequences. As each sequence is depleted, the entry to this routine is at USTOP_n (to close the compare loop for the particular sequence) instead of to this point.

The starting address of the input area for the current sequence (ABEGIN, BBEGIN, etc.) is used along with the corresponding logical unit address to initialize a channel program to get a block of records from disk storage.

RDABCD, CD-F2

An EXCP macro is issued and a block of records is read into the specified input area in main storage.

For all sequences except the highest one (G in a 7-way merge or D in a 4-way merge), the routine continues to LM1 (or LM2, etc., as the case may be). When the highest sequence is being processed, the routine branches directly to IRMDR.

LM1, CD-G3

This location is a switch that will be on (no-op) only during the last merge of a pass. Until such time, the routine branches to IRMDR.

During the last merge of a pass, a test is made to determine if the input interleave factor for the current sequence should be reduced. This factor is reduced when the number of blocks processed in the current sequence is one greater than the number of blocks in the last sequence of a pass (LSTRG_n). The count in LSTRG_n is reduced by one each time this function is entered during the last merge of a pass. If the decremented count is equal to or higher than zero, a branch is made to IRMDR; if lower, the branch is to IRMDR1.

This location is entered when the input interleave factors are not to be changed. The factors are stored in RMDR and QUOT and the program continues to the routine to calculate the next interleaved disk address at CALADR (Chart CM). The program returns to this routine at BYPAS1+4.

IRMDR1, CD-H5

This location is entered when the input interleave factors need to be changed. The reduced factors for the next lower order of merge are stored in RMDR and QUOT and a branch is made to the routine to calculate the next interleaved disk address at CALADR (Chart CM). The program returns to this routine at BYPAS1+4.

BYPAS1+4, CD-J3

The current interleaved disk address (CHHR) is stored in ARADDR (or BRADDR, etc., as the case may be) and a branch is made to an address that varies according to the current sequence and the order of merge. These addresses are listed in the table on Chart CD.

SEQUENCE G COMPARE LOOP, FIXED-LENGTH RECORDS - CE

The program that was loaded into main storage at the start of phase 2 was for either a 7-way (DSORT202) or a 4-way merge (DSORT201). For a 4-way merge, the compare loops start at sequence D (Chart CH).

For a 7-way merge, the input routine (Chart CD) initialized certain branch instructions in all the compare loops from G through B. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the sequences being merged. As each sequence is depleted, a branch is made to the compare loop for the next lower order of merge (Charts CF through CK, consecutively). A branch-and-link to the output routine (Chart CL) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence G, F, or E. It then exits to the sequence F compare loop (Chart CF).

The exit from this loop is provided at the beginning so that no processing need be done when sequence G is depleted. If such is the case, a branch is made to COMPFE in the sequence F compare loop (Chart CF); if not, a record from sequence G is compared with a record from sequence F. As long as G is determined to be the winner, it is compared with records from the other available sequences in turn. The routine thus continues in this compare loop or exits to another loop, depending on the results of each comparison. For example, when merging in ascending sequence:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMPGF	G F	COMPGE COMPFE (Chart CF)
COMPGE	G E	COMPGD COMPED (Chart CG)
COMPGD	G D	COMPGC COMPDC (Chart CH)
COMPGC	G C	COMPGB COMPGB (Chart CJ)
COMPGB	G B	COMPGA COMPBA (Chart CK)
COMPGA	G A	PUTG PUTA (Chart CK)

The branch exits are determined not only by the results of the comparison but also by the depletion of input sequences. For example, if in COMPGE sequence E is found to be depleted, the instruction at BGD is an unconditional branch to COMPGD to compare the G record with the next D record.

Another variation in the compare loop operation occurs when, for example, a G record is found to be the winner through COMPGF and COMPGE. Then, in COMPGD, the D record is found to be the winner. The exit from the loop, as previously described, is to COMPDC (Chart CH); however, the return address that is saved in register SAVED is COMPGD+2. Then, assuming the D record is the winner through compare loops C, B, and A, it is moved to the output area, and control is returned to the G loop at the address in SAVED. The reason for entering this loop at COMPGD+2 is that although D was the winner, the G record had already been determined to be winner over F and E at that time. Therefore, the comparing in G loop resumes at the point where G is compared with the next D record.

PUTG, CE-J2

The address of the winning record from sequence G is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CL). Control is returned to this routine at NXTGR.

NXTGR, CE-H4

The address for the sequence G input area is updated and, if there are more records in the area, a branch is made back to COMPGF to compare the next record. If, however, there are no more records in the G input area, a test is then made to determine if the end of input sequence G has been reached (hexadecimal EF in the last byte of the block). If so, a branch is made to USTOPG in the input routine (Chart CD) to close the G compare loop so that future entries during this merge will branch directly from COMPGF to COMPFE. If the G sequence is not yet depleted, a branch is made to GETG in the input routine to refill the input area.

SEQUENCE F COMPARE LOOP, FIXED-LENGTH RECORDS - CF

For a 6-way merge, the input routine (Chart CD) initialized certain branch instructions in all the compare loops from F through B. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the sequences being merged. As each sequence is depleted, a branch is made to the compare loop for the next lower order of merge (Charts CG through CK, consecutively). A branch-and-link to the output routine (Chart CL) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence F. It then exits to the sequence E compare loop (Chart CG).

COMPFE, CF-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence F is depleted. If such is the case, a branch is made to COMPED in the sequence E compare loop (Chart CG); if not, a record from sequence F is compared with a record from sequence E. As long as F is determined to be the winner, it is

compared with records from the other available sequences in turn. The routine thus continues in this compare loop or exits to another loop, depending on the results of each comparison. For example, when merging in ascending sequence:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMPFE	F E	COMPFD COMPED (Chart CG)
COMPFD	F D	COMPFC COMPDC (Chart CH)
COMPFC	F C	COMPFB COMPBC (Chart CJ)
COMPFB	F B	COMPFA COMPBA (Chart CK)
COMPFA	F A	PUTF PUTA (Chart CK)

The branch locations are determined not only by the results of the comparison but also by the depletion of input sequences. For example, if in COMPFD sequence D is found to be depleted, the instruction at BFC is an unconditional branch to COMPFC to compare the F record with the next C record.

Another variation in the compare loop operation occurs when, for example, an F record is found to be the winner through COMPFE and COMPFD. Then, in COMPFC, the C record is found to be the winner. The exit from the loop, as previously described, is to COMPBC (Chart CJ); however, the return address that is saved in register SAVEC is COMPFC+2. Then, assuming the C record is the winner through compare loops B and A, it is moved to the output area, and control is returned to the F loop at the address in SAVEC. The reason for entering this loop at COMPFC+2 is that although C was the winner, the F record had already been determined to be winner over E and D at that time. Therefore, the comparing in F loop resumes at the point where F is compared with the next C record.

PUTF, CF-H2

The address of the winning record from sequence F is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CL). Control is returned to this routine at NXTGR.

NXTFR, CF-H4

The address for the sequence F input area is updated and, if there are more records in the area, a branch is made back to the start of the highest available compare loop (COMPGF or COMPFE) to compare the next record. If, however, there are no more records in the F input area, a test is made to determine if the end of input sequence F has been reached (hexadecimal EF in last byte of block). If so, a branch is made to USTOPF in the input routine (Chart CD) to close the F compare loop so that future entries during this merge will branch directly from COMPFE to COMPED. If the F sequence is not yet depleted, a branch is made to GETF in the input routine to refill the input area.

SEQUENCE E COMPARE LOOP, FIXED-LENGTH RECORDS - CG

For a 5-way merge, the input routine (Chart CD) initialized certain branch instructions in all the compare loops from E through B. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the sequences being merged. As each sequence is depleted, a branch is made to the compare loop for the next lower order of merge (Charts CH through CK, consecutively). A branch-and-link to the output routine (Chart CL) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence E. It then exits to the sequence D compare loop (Chart CH).

COMPED, CG-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence E is depleted. If such is the case, a branch is made to COMPDC in the sequence D compare loop (Chart CH); if not, a record from sequence E is compared with a record from sequence D. As long as E is determined to be the winner, it is compared with records from the other available sequences in turn. The routine thus continues in this compare loop or exits to another loop, depending on the results of each comparison. For example, when merging in ascending sequence:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMPED	E	COMPEC
	D	COMPDC (Chart CH)
COMPEC	E	COMPEB
	C	COMPBC (Chart CJ)
COMPEB	E	COMPEA
	B	COMPBA (Chart CK)
COMPEA	E	PUTE
	A	PUTA (Chart CK)

The branch locations are determined not only by the results of the comparison but also by the depletion of input sequences. For example, if in COMPEC sequence C is found to be depleted, the instruction at BEB is an unconditional branch to COMPEB to compare the E record with the next B record.

Another variation in the compare loop operation occurs when, for example, an E record is found to be the winner through COMPED and COMPEC. Then, in COMPEB, the B record is found to be the winner. The exit from the loop, as previously described, is to COMPBA (Chart CK); however, the return address that is saved in register SAVEB is COMPEB+2. Then, assuming the B record is the winner through compare loop B, it is moved to the output area, and control is returned to the E loop at the address in SAVEB. The reason for entering this loop at COMPEB+2 is that although B was the winner, the E record had already been determined to be winner over D and C at that time. Therefore, the comparing in E loop resumes at the point where E is compared with the next B record.

PUTE, CG-G2

The address of the winning record from sequence E is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CL). Control is returned to this routine at NXTFR.

NXTER, CG-H4

The address of the sequence E input area is updated and, if there are more records in the area, a branch is made back to the start of the highest available compare loop (COMPGF, COMPFE, or COMPED) to compare the next record. If, however, there are no more records in the E input area, a test is made to determine if the end of input sequence E has been reached (hexadecimal EF

in last byte of block). If so, a branch is made to USTOPE in the input routine (Chart CD) to close the E compare loop so that future entries during this merge will branch directly from COMPED to COMPDC. If the E sequence is not yet depleted, a branch is made to GETE in the input routine to refill the input area.

SEQUENCE D COMPARE LOOP, FIXED-LENGTH RECORDS - CH

This compare loop is entered:

- In a 7-way merge, when the record from sequence D is found to be the winner in a previous compare loop.
- In a 7-way merge, when sequences G, F, and E have been depleted.
- In a 4-way merge, at the completion of the input routine.

For a 4-way merge, the input routine (Chart CD) initialized certain branch instructions in all the compare loops from D through B. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the sequences being merged. As each sequence is depleted, a branch is made to the compare loop for the next lower order of merge (Charts CJ and/or CK). A branch-and-link to the output routine (Chart CL) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence D. It then exits to the sequence C compare loop (Chart CJ).

COMPDC, CH-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence D is depleted. If such is the case, a branch is made to COMPCB in the sequence C compare loop (Chart CJ); if not, a record from sequence D is compared with a record from sequence C. As long as D is determined to be the winner, it is compared with records from the other available sequences in turn. The routine thus continues in this compare loop or exits to another loop, depending on the results of each comparison. For example, when merging in ascending sequence:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMPDC	D	COMPDB
	C	COMPCB (Chart CJ)
COMPDB	D	COMPDA
	B	COMPBA (Chart CK)
COMPDA	D	PUTD
	A	PUTA (Chart CK)

The branch locations are determined not only by the results of the comparison but also by the depletion of input sequences. For example, if in COMPDB sequence B is found to be depleted, the instruction at BDA is an unconditional branch to COMPDA to compare the D record with the next A record.

Another variation in the compare loop operation occurs when, for example, a D record is found to be the winner in COMPDC. Then, in COMPDB, the B record is found to be the winner. The exit from the loop, as previously described, is to COMPBA (Chart CK); however, the return address that is saved in register SAVEB is COMPDB+2. Then, assuming the B record is the winner in compare loop A, it is moved to the output area, and control is returned to the D loop at the address in SAVEB. The reason for entering this loop at COMPDB+2 is that although B was the winner, the D record had already been determined to be winner over C at that time. Therefore, the comparing in D loop resumes at the point where D is compared with the next B record.

PUTD, CH-F2

The address of the winning record from sequence D is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CL). Control is returned to this routine at NXTDR.

NXTDR, CH-H3

The address for the sequence D input area is updated and, if there are more records in the area, a branch is made back to the start of the highest available compare loop (depending upon the order of merge) to compare the next record. If, however, there are no more records in the D input area, a test is made to determine if the end of input sequence D has been reached (hexadecimal EF in last byte of block). If so, a branch is made to USTOPD in the input routine (Chart CD) to close the D compare loop so that future entries during this

merge will branch directly from COMPDC to COMPCB. If the D sequence is not yet depleted, a branch is made to GETD in the input routine to refill the input area.

SEQUENCE C COMPARE LOOP, FIXED-LENGTH RECORDS - CJ

For a 3-way merge, the input routine (Chart CD) initialized certain branch instructions in compare loops C and B. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the sequences being merged. As each sequence is depleted, a branch is made to the compare loop for the next lower order of merge (Chart CK). A branch-and-link to the output routine (Chart CL) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence C. It then exits to the sequence B compare loop (Chart CK).

COMPCB, CJ-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence C is depleted. If such is the case, a branch is made to COMPBA in the sequence B compare loop (Chart CK); if not, a record from sequence C is compared with a record from sequence B. If the C record is the winner, it is compared with a record from sequence A. If the C record wins again, the routine continues to PUTC.

The other branch locations, in the event that either B or A is determined to be the winner, are:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMPCB	B	COMPBA (Chart CK)
COMPCA	A	PUTA (Chart CK)

The branch locations are determined not only by the results of the comparison but also by the depletion of input sequences. For example, if in COMPCA sequence A is found to be depleted, the instruction at BPUTC is an unconditional branch to PUTC to prepare to move the C record to the output area.

Another variation in the compare loop occurs when, for example, a C record is found to be the winner in COMPCB. Then, in COMPCA, the A record is found to be the winner. The exit from the loop, as

previously described, is to PUTA (Chart CK); however, the return address that is saved in register SAVEA is COMPCA+2. Then, after the A record is moved to the output area, control is returned to the C loop at the address in SAVEA. The reason for entering this loop at COMPCA+2 is that although A was the winner, the C record had already been determined to be winner over the B record. Therefore, the comparing in C loop resumes at the point where C is compared with the next A record.

PUTC, CJ-E2

The address of the winning record from sequence C is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CL). Control is returned to this routine at NXTCR.

NXTCR, CJ-G3

The address for the sequence C input area is updated and, if there are more records in the area, a branch is made back to the start of the highest available compare loop (depending on the order of merge) to compare the next record. If, however, there are no more records in the C input area, a test is made to determine if the end of input sequence C has been reached (hexadecimal EF in last byte of block). If so, a branch is made to USTOPC in the input routine (Chart CD) to close the C compare loop so that future entries during this merge will branch directly from COMPCB to COMPBA. If the C sequence is not yet depleted, a branch is made to GETC in the input routine to refill the input area.

SEQUENCE B COMPARE LOOP, FIXED-LENGTH RECORDS - CK

For a 2-way merge, the input routine (Chart CD) initialized certain branch instructions in compare loop B. However, the flow through this loop varies not only with the order of merge but also, later on, with the depletion of records in sequence B. When sequence B is depleted, a branch is made to prepare to move the A record to the output area.

A branch-and-link to the output routine (Chart CL) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from

sequence B. It then branches directly to PUTA.

COMPBA, CK-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence B is depleted. If such is the case, a branch is made to PUTA; if not, a record from sequence B is compared with a record from sequence A. If the B record is the winner, the routine branches to PUTB; if the A record is the winner, the routine branches to PUTA.

PUTB, CK-D2

The address of the winning record from sequence B is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CL). Control is returned to this routine at NXTBR.

NXTBR, CK-G2

The address for the sequence B input area is updated and, if there are more records in the area, a branch is made back to the start of the highest available compare loop (depending on the order of merge) to compare the next record. If, however, there are no more records in the B input area, a test is made to determine if the end of input sequence B has been reached (hexadecimal EF in the last byte of block). If so, a branch is made to USTOPB in the input routine (Chart CD) to close the B compare loop so that future entries during this merge will branch directly from COMPBA to PUTA. If the B sequence is not yet depleted, a branch is made to GETB in the input routine to refill the input area.

PUTA, CK-D4

The address of the winning record from sequence A is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CL). Control is returned to this routine at NXTAR.

NXTAR, CK-G4

The address for the sequence A input area is updated and, if there are more records in the area, a branch is made back to the start of the highest available compare loop (depending on the order of merge) to compare the next record. If, however, there are no more records in the A input area, a test is made to determine if the end of input sequence A has been reached (hexadecimal EF in last byte of block). If so, a branch is made to USTOPA in the input routine (Chart CD) to close the sequence A compares. If the A sequence is not yet depleted, a branch is made to GETA in the input routine to refill the input area.

OUTPUT ROUTINE, FIXED-LENGTH RECORDS - CL

This routine is entered at one of three locations:

- OUTFUL, when a winning record has been found in one of the compare loops.
- MMPP, when an end-of-merge condition is detected in the input routine (end of current set of input sequences).
- MOV MVC, (via STR2), when a winning record is to be moved immediately after writing a block on disk (the output area was full).

When the routine is entered at OUTFUL, and the output area is not full, the winning record is moved to the output area and the program returns to the compare loop in which the winning record was found. When the output area is full, the contents are first written in the output portion of the disk work area, the output interleave factor is reduced, if necessary, and the new disk address is calculated (Chart CM). The winning record is then moved to the output area and the program returns to the compare loop in which the winning record was found.

When the routine is entered at MMPP, an end-of-sequence indicator (hexadecimal EF) is inserted in the last byte of the current output block and the end-of-merge switch is turned on. The last output block is then written out, the output interleave factor is changed, if necessary, and the new disk address is calculated (Chart CM). The program then branches to the merge-merge routine (Chart CB).

OUTFUL, CL-C2

The current address of the output area (in register PUTOUT) is compared to the address of the end of the output area (OUTEND). If the output area is full, a branch is made to WRITE; if not, a branch is made to the move routine via the selector table (STR2).

MOV MVC, CL-D2

The winning record (address in MREG) is moved to the output area. This move subroutine is entered via STR2; the move instructions were initialized so that only the actual number of bytes are moved at execution time. As part of the initialization, a branch exit to ZYXWZY was placed at the end of the move instruction(s).

ZYXWZY, CL-E2

The current address of the output area (register PUTOUT) is updated and the routine branches back through the LINK register to the compare loop in which the winning record was determined. The possible addresses are listed at the off-page connector from block E2 on Chart CL.

MMPP, CL-C3

The end-of-sequence indicator (hexadecimal EF) is inserted at the end of the block currently in the output area and the end-of-merge switch (MMPP1 on Chart CM) is turned on. The routine then continues to WRITE.

WRITE, CL-D3

The address of the start of the output area (OBEGIN) is restored in register PUTOUT, and the current disk address of the output sequence is moved to the current disk interleave address. If the BPT (blocks per track) is greater than 1, a branch is made to WTDATA; if BPT = 1 (a condition that can occur only during pass 1), the output count field address is stored, the current interleave address is reduced by one, and the output area address in register PUTOUT is incremented by eight to put it past the

count field. The routine then continues to WTDATA.

WTDATA, CL-E3

An EXCP macro is issued and the block of records is written from the main storage output area into the output portion of the disk work area. The parameters for the write operation are supplied by the command control block at OCCB. The routine then continues to LMO.

LMO, CL-F3

This location is a switch that will be on (no-op) only when the remaining input sequences are equal to or less than the order of merge squared ($S \leq M^2$). This condition is detected in the merge-merge routine (see LMO SW, Chart CB-F3). Until such time, LMO is a branch to ORMDR.

When $S \leq M^2$, a calculation and test are made to determine if the number of blocks merged to the output sequence is greater by one than the number of blocks contained in the last merge of the pass. If it is, the output interleave factors are to be reduced and a branch is made to ORMDR1; otherwise, the routine continues to ORMDR.

ORMDR, CL-H3

This location is entered when the output interleave factors are not to be changed. The factors are stored in RMDR and QUOT and the program branches to the routine to calculate the next interleaved disk address at CALADR (Chart CM). Upon return to the output routine (when switch MMPP1 is off), the entry point is STR2.

ORMDR1, CL-H4

This location is entered when the output interleave factors need to be changed. The reduced factors for the next lower order of merge are stored in RMDR and QUOT and the program branches to the routine to calculate the next interleaved disk address at CALADR (Chart CM). Upon return to the output routine (when switch MMPP1 is off), the entry point is STR2.

CALCULATE INTERLEAVED DISK ADDRESS ROUTINE,
FIXED-LENGTH RECORDS - CM

This routine is entered from the input routine (Chart CD) whenever a block is read, or from the output routine (Chart CL) whenever a block is written.

The current interleaved disk address is updated with the factors in RMDR and QUOT, which have been placed there by the routine (input or output) immediately before entry to this routine.

The record number and head/track number are calculated and checked for validity. If maximums are exceeded, the next higher valid address is calculated. The upper limit of the current work area extent is then checked and, if exceeded, a new address is calculated based on the lower limit of the next logical segment of the work area.

See Figure 35 for an illustration of interleaved disk address calculations.

CALADR, CM-C2

The interleave factor in RMDR is used to calculate the next record number by adding it to the current disk address (CHHR) in register 0. If the new record number is greater than the maximum number of sort blocks per track (BPT), it is not valid; the next valid record number is then calculated by adding the 256-complement of the BPT (BCOMP) to the value in register 0.

The interleave factor in QUOT is then used to calculate the new head/track number by adding it to the newly-calculated record number in register 0. If the new head/track number exceeds nine, it is not valid; the next valid cylinder/head number is calculated by adding the 256-complement of ten (HCOMP) to the value in register 0.

A test is then made to determine if the upper limit of the current work area extent has been exceeded by the new disk address just calculated in register 0. If not, the new address is valid and a branch is made to LMTSOK; if so, the disk address is re-calculated based on the lower limit of the next extent in the work area.

LMTSOK, CM-E3

The new interleaved disk address, for input or output, as the case may be, is stored in CHHR.

BSTR6, CM-F3

If the interleaved address just calculated and stored was for output, it is moved from CHHR to ORADDR and the routine continues to MMPP1. If the address was for input, a branch is made to BYPAS1+4 (or BYPAS2+4, etc., as the case may be) in the input routine (Chart CD), where the address will be moved from CHHR to ARADDR (or BRADDR, etc., as the case may be).

MMPP1, CM-H3

This location is normally a branch back to MOV MVC (via STR2) in the output routine (Chart CL) to move a winning record that could not be moved earlier because the output area was full. At the end of a merge, however, MMPP1 would have been changed to a no-op (at MMPP in the output routine) so that the program will continue to MMPP2 in the merge-merge routine (Chart CB).

PHASE 2 INITIALIZATION, VARIABLE-LENGTH
RECORDS - CN

The checkpoint record, created by the assignment phase and updated by phase 1, is read into main storage to obtain information to be used in phase 2. This routine then initializes:

- Merge-merge routine
- Pass-pass routine
- Sequence compare loops
- Interleave address routine
- Relocatable routines
- Constants

Order of Merge -- 4
 Blocks Per Track -- 5

RMDR -- 0004
 QUOT -- 0000
 BCOMP -- 00FB

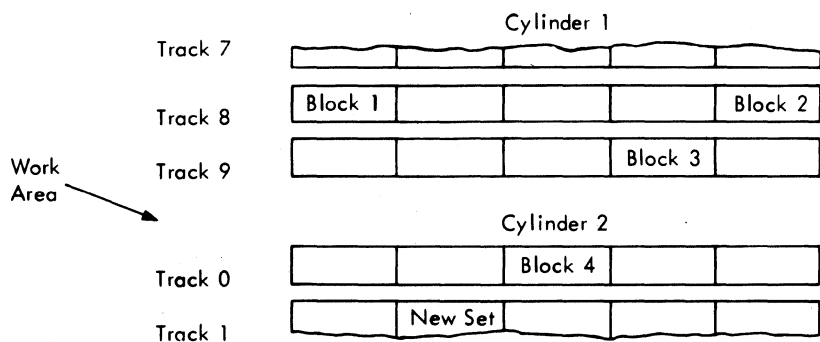
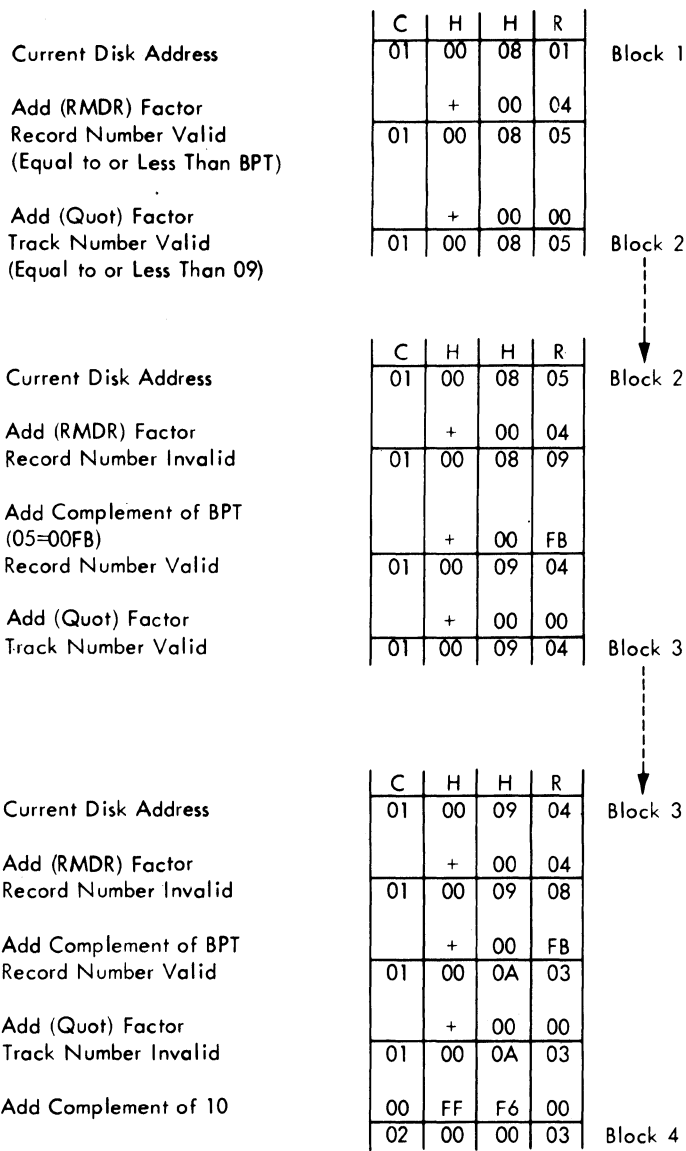


Figure 35. Calculate Interleaved Disk Address

Merge-merge. Initialized according to the order of merge that was calculated by the assignment phase (from 2 to 6).

Pass-pass. Initialized according to the order of merge to be used during phase 2. Input and output interleave factors, as well as pointers to the logical halves of the work area table, are stored at this time.

Compare. All compares within the compare loops are initialized with the length and location of the first control data field. Branches are initialized for either ascending or descending sequence.

Interleave Address routine. Factors are calculated for the input/output disk address interleaving for phase 2.

Relocatable routines. The equal routine is not relocated. However, if the equal routine is not required or if the records contain less than 12 control fields, it will be bypassed or shortened, as the case may be. When control is returned by the relocater, initialization continues and the pass-pass routine is written out on the 2311 checkpoint track as record two. Subsequently, it is brought in to main storage only when an end-of-pass condition exists.

The next step is to calculate or allocate the input/output areas to be used in the merging process. The number of main storage areas needed is equal to the order of merge plus one. Each area is equal to the sort-block size. Each input area also has an overflow area which is LMAX-1 in length (maximum logical record length minus one). The input/output channel programs are then initialized with the number of bytes to be transferred during an I/O operation (number of bytes is equal to the sort-block size). The program then continues to the merge-merge routine (Chart CP).

INTPH2, CN-B2

The base register (register 11) is loaded with the starting address of phase 2. This address is obtained from CKPCCB+8. Registers 2 and 3, which were loaded at the end of phase 1, are stored:

<u>Reg.</u>	<u>Contents</u>	<u>Stored in</u>
2	Logical unit address of device where checkpoint is written.	1. CCB for taking checkpoint at every pass (RSTCCB). 2. CCB for reading checkpoint (CKPCCB). 3. CCB for reading and writing the pass-pass routine (PPCCB).
3	Disk address of checkpoint record.	1. Location CHECKP, for updating checkpoint record and for passing this address to phase 3. 2. Location CHHR, current disk interleave address. Used only to read the checkpoint record for initialization.

RCHKPT, CN-C2

An EXCP macro is issued and the checkpoint record is read into location PASSNO. When the read operation is completed, the routine continues to RDCPOK.

RDCPOK, CN-D2

A test is made to determine if the equal routine is required (more than one control field per record). If so, a branch is made to COMPIT; if not, the branches in the mainline compare loops are initialized to bypass the branch-and-link to the equal routine when two equal records are encountered.

COMPIT, CN-F2

The length and displacement (location within record) of the first control field are OR'ed into the compare instructions in the mainline compare loops. The instructions that determine the branching conditions in the compare loops are then initialized for ascending or descending sequence.

The order of merge (M) and blocks per track (BPT1) are obtained from the checkpoint record and stored in constants PH2IOM and BPT, respectively. Registers are then initialized for calculating interleave factors for disk addresses.

DIVAGN, CN-G2

The interleave factors are calculated by dividing the order of merge by the sort blocks per track. The quotient is stored in location NQOUT and the remainder in location NRMDR. The order of merge is then reduced by one and the next factors are calculated and stored adjacent to the first. This process is repeated for a number of times that is equal to the order of merge. These factors will be used by the interleaved disk address routine; the quotients for calculating the head or track numbers, the remainders for calculating the record numbers.

The merge-merge routine is initialized with a switch (at WAY4) for the specified order of merge for phase 2.

The pass-pass routine is then initialized with indexing factors (from POINTI) that point to the logical halves of the work area table.

The number of blocks per sequence is then obtained from the checkpoint record and stored at OUTPTG.

A test is made to determine if a copy pass is required in phase 2. If the user has specified that the output file is to be written in the first half of the disk work area, and if an inaccurate file size has been specified on the SORT control statement, the input for phase 3 could be in the first half of the work area (based on the number of passes calculated by phase 1). When this condition exists, an extra pass must be made so that the phase 3 input will be in the second half of the work area.

OUP TOK, CN-D3

The physical limits of the work area are obtained from the checkpoint record (LOGPHY) and placed in the LIMITS table. A branch is then made to the relocater routine (Chart FA). After the equal routine is included (if required) and its length is determined, control is returned to this routine at START.

START, CN-F3

The pass-pass routine is initialized with the starting address of the phase 2 input and output areas in main storage. This address, which will later be the address into which the checkpoint record will be read, is obtained from location RLISA after the relocater routine is executed.

CPIOAS, CN-B4

An overflow area, equal in size to the maximum logical record length minus one (LMAX-1), is allocated adjacent to the beginning of each input area. These areas are used for compacting split records when they are encountered (see Figure 29).

Constants for phase 2 input and output areas are now calculated. These are:

- ABEGIN through FBEGIN - Starting addresses of input areas A, B, C, D, E, and F (as required by order of merge).
- AEND through FEND - Ending address of input areas A, B, C, D, E, and F (as required by order of merge).
- OBEGIN - Starting address of output area.

If the sort blocks per track (BPT) is equal to 1, the second half of the work area was not formatted by phase 1. When such is the case, instructions are now modified so that pass 1 of phase 2 will format the second half of the disk work area, which is the pass 1 output portion.

The pass-pass routine is then written on the checkpoint track as record 2.

BYWCKD, CN-F4

This instruction is a no-op except when the BPT is greater than 1, in which case a branch is made to NOWCKD. When BPT = 1 and the format must be established for the output portion of the disk work area, the routine allocates the output count field next to the output area and increases the output block count by eight.

The ending address of the output area (OUTEND) is calculated and stored. The read and write CCW's are then initialized with the input and output data counts and a branch is made to the merge-merge routine (Chart CP).

MERGE-MERGE ROUTINE, VARIABLE-LENGTH RECORDS - CP

The merge-merge routine initializes the mainline to merge the next set of input sequences into one output sequence. A set is equal in number to the OM (order of merge) calculated by the assignment phase, except during the last merge of a pass when a set may have less sequences than the OM.

During phase 1, fixed-length blocks were created to include the variable-length records. It is possible, therefore, that the last block of an output sequence from phase 1 will not be full. Phase 2, in all passes except the first, would most likely decrease the number of blocks in an output sequence in such cases. For example, if phase 2 is initialized to merge 2 sequences of three blocks each into an output sequence, the total output blocks should equal 6. Because the last blocks of the two sequences might not be full, the six blocks could possibly fit into five output blocks when merged in phase 2. To keep the interleaving technique the same as for fixed-length records, the address of the sixth block must be calculated and stored in the output address table even though the sixth block may be empty.

The functions performed by this routine vary with the conditions at time of entry. These conditions are:

- End-of-pass, which occurs (1) at the start of phase 2 and (2) each time all the input sequences have been merged into output sequences. For example, if phase 1 output is 64 sequences and the OM is 4, an end-of-pass condition would exist (1) at the start of phase 2, (2) when 64 input sequences have been merged into 16 output sequences, and (3) when the 16 new input sequences (the output from the previous pass) have been merged into 4 new output sequences.
- End-of-merge with more than M² input sequences remaining.
- End-of-merge with M² or less input sequences remaining.

- End-of-merge with M or less input sequences remaining (which signifies that the upcoming merge is the last one of the current pass).

The functions performed for each condition are:

End-of-pass. The last-merge switches in the input and output routines are turned off and a branch is made to the pass-pass routine. When control is returned to this routine, there would no longer be an end-of-pass condition; at this point, it would be either (1) end-of-merge with more than M² sequences or (2) end-of-merge with M² or less sequences. These functions are described in the following paragraphs under their respective headings. Note that the only function that will not be performed in an end-of-pass condition is the shifting of the output disk address table. This table is never shifted before the first merge of a pass.

End-of-merge with more than M² sequences. The mainline is re-initialized for the order of merge calculated by the assignment phase and the number of input sequences is reduced by a number equal to the order of merge. The number of merges to be done before new output interleave factors are implemented is reduced by one. Then the output disk address table is shifted to obtain the starting address for the next output sequence.

End-of-merge with M² or less sequences. If this is the last merge of a pass, the functions performed are described in the next paragraph. If not the last merge, the mainline is initialized with new interleave factors to be used in computing the disk output addresses for the remainder of the pass. The output sequences created from this point on will become the input sequences for the last merge of the next pass.

End-of-merge with M or less sequences. When this condition is reached, it is the last merge of a pass. The input interleave factors are changed and the end-of-pass switch is turned on. The program will then continue to the pass-pass routine at the completion of the last merge.

MMPP2, CP-B3

The end-of-merge switch (MMPP1 in the interleave factors routine, Chart CY) is turned off and remains off until next time an end-of-merge condition is detected. Turning off switch MMPP1 consists of activating the branch to the input routine.

MMPPS, CP-C3

If this routine is being entered on an end-of-pass condition, a branch is made to LM1234. For an end-of-merge condition, the end-of-sequence indicator (hexadecimal F0) is cleared from the end of the output area and registers are initialized to open the mainline and fill the input areas (registers SAVEA through SAVEC, depending on OM).

TMINUS, CP-C4

At the end of every merge in a pass (except the last) a check is made to see if the input addresses (ARADDR-FRADDR) and the output address (ORADDR) are updated to the disk address of the first block for the next set of M sequences. If an address is not updated (because the previous input or output sequence did not contain the number of blocks that it should), the address is updated in the following manner:

A new address is computed until the sequence block count associated with the address becomes zero. Every time an address is computed for a sequence during a merge, the sequence block count for that sequence is decremented by one. At the completion of a merge, all sequence block counts will be zero if all sequences contained the right number of blocks. If any sequences were short, the block counts will be positive. Whenever a sequence has a block count that is not zero, the address for the first block of the next respective sequence must be updated.

When updating has been completed, if necessary, each address (ARADDR-FRADDR and ORADDR) is re-initialized with new sequence block counts for the next merge. See Figure 36 for an illustration of the updating procedure just described.

The input sequence counter (SR) is tested to determine if the next merge will be the last one of the pass ($SR \leq OM$). If not, a branch is made to WAY4; if so, the routine continues to REDUCEI.

REDUCEI, CP-C2

Instructions at IRMDR and IQUOT are initialized with new input interleave factors for the last merge. The

end-of-pass switch at MMPPS is turned on (it will be turned off for the next pass in the pass-pass routine) and the routine continues to LM1234.

LM1234, CP-D3

One of two courses of action is taken at this point:

- End of merge - Turn on last merge input interleave switches (LM1 through LM_n, depending on OM) and continue to WAY4-4.
- End of pass - Turn off switches LM1 through LM_n and branch to LMOSW.

At WAY4-4, the pass-pass routine has inserted an instruction that initializes the mainline with the order of merge that is to be used during the last merge of a pass.

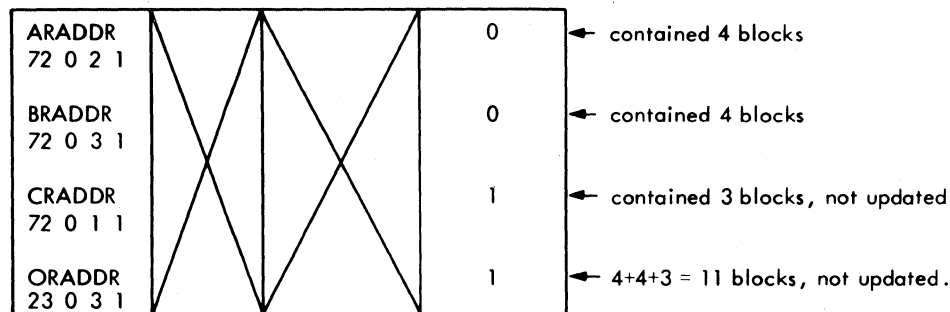
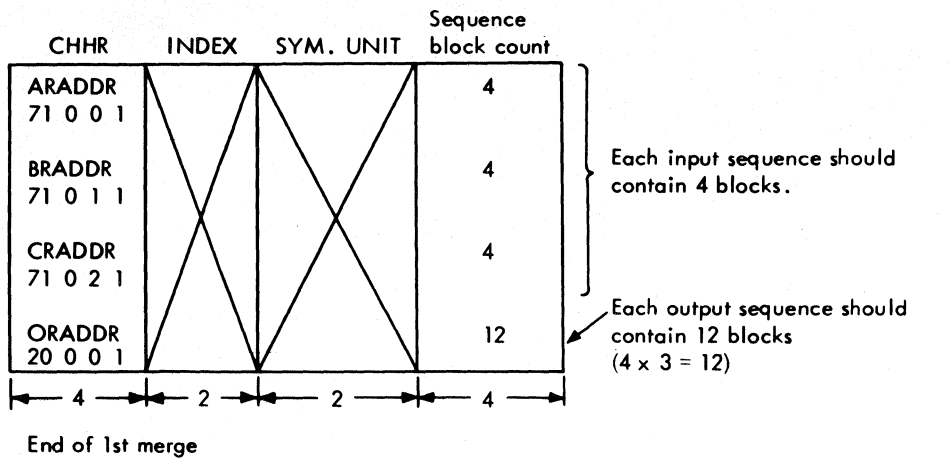
WAY4, CP-E4

The input sequence counter (SR) is decremented by a factor equal to the order of merge. Location MAXFAC is then tested to determine if the output interleave factors are to be changed (see Appendix A for description of MAXFAC contents). When the count in MAXFAC is at zero, it signifies that M^2 or less input sequences remain to be merged. Under these conditions, new output interleave factors are to be implemented and the routine continues to REDUCEO. As long as the count in MAXFAC is not zero, the count is decremented by one and the routine branches to NOCHG.

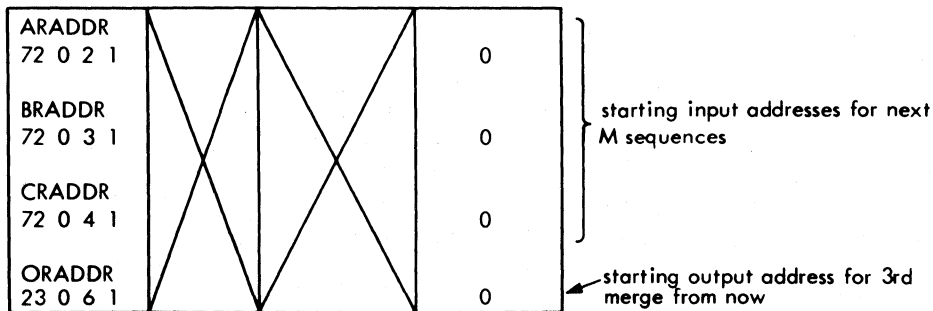
REDUCEO, CP-E2

Instructions at ORMDR and OQUOT are initialized with new output interleave factors, which have been pre-determined by the pass-pass routine. Continue to LMOSW.

EXAMPLE: 3 Way Merge, 1 Block Per Track



After 1MINUS updating



Remember the output address table must be shifted; therefore starting output disk address is 20.0.1.1

Figure 36. Variable-Length Disk Address - Block Count

LMOSW, CP-F3

At the end of a pass, the output interleave switch is turned off (LMO in the output routine, Chart CX). This switch had been turned on (branch 00) during the pass when the number of input sequences remaining was equal to, or less than, M^2 (when the count in MAXFAC was reduced to zero). After switch LMO is turned off, a branch is made

to ENDPAS if it is the end of a pass; otherwise, the routine continues to NOCHG.

ENDPAS, CP-H3

At the end of each pass, as well as at the start of phase 2 (which is considered an end-of-pass condition), the pass-pass

routine is read into the main storage I/O areas by channel program PPCHPG. The program then branches to EXECPP in the pass-pass routine, executes the routine, and returns control to this point in the merge-merge routine. The channel program then writes the pass-pass routine back on the checkpoint track in the work area and a branch is made to MMPPS+4 to start merging in the new pass.

NOCHG, CP-H4

For each new merge within a pass, the count for MAXFAC (which was just decremented at REDUCEO-4) is stored back in MAXFAC. Except for the first merge of every pass, the output disk address table (ORADDR) is shifted to give the starting disk address for the new output sequence. The output sequence block counter (LMSTRG) is re-initialized with a count equal to the number of blocks contained in the sequences which comprise the last merge of the pass. This counter is used at LMO in the output routine to detect when the output interleave factors should be reduced. The program then branches to USTOPA to open the mainline compare routine for sequence A.

PASS-PASS ROUTINE, VARIABLE-LENGTH RECORDS
- CQ

At the beginning of each phase-2 pass, the pass-pass routine is read from the checkpoint track into the main storage input/output areas at location EXECPP. The routine first reads the checkpoint record, updates it, and restores it to the checkpoint track. It then computes the initial disk addresses for M sequences for both input and output. The two pointers to the logical halves of the work area table are reversed so that initial disk addresses can be computed at the beginning of the next pass. Next, the interleave factors are restored to their original values which were computed during phase 2 initialization (order of merge divided by BPT). The number of the pass being entered is listed on SYSLOG. If the upcoming pass is the last one, a switch is set to enable job control to fetch phase 3 into main storage.

The pass-pass routine then calculates the interleave factors to be used (1) during the last merge of the pass for input and (2) during the last set of merges for output (when M² or less input sequences are merged into M or less output sequences). The merge-merge routine is initialized with a switch that determines the order of merge during the last merge of the pass.

At the end of the pass-pass routine, control is returned to the merge-merge routine which writes the pass-pass routine back on the checkpoint track (record 2).

EXECPP, CQ-B2

The pass-pass routine is entered each time the entire file has been passed through or merged into a new set of sequences. It initializes phase 2 for the upcoming pass (except the last).

The end-of-pass switch (MMPPS in the merge-merge routine), is turned off (branch 00) and the routine initializes to read the checkpoint record into storage.

TAKECP, CQ-C2

The checkpoint record is read into the first 316 bytes of the phase 2 input/output areas.

CKPTOK, CQ-D2

The checkpoint record is updated with:

1. A decimal integer representing the phase 2 pass number (PH2PAS)
2. The number of passes remaining (NOPASS)
3. The number of sequences to be merged during the upcoming pass (NSR)
4. The number of sort blocks that comprise the last merge of the pass (LMBLOK)
5. The number of sort blocks contained in the last sequence of the pass (MERGEL)
6. The number of sort blocks which should be contained in a given sequence (OUTPTG)
7. Two hexadecimal pointers that reflect the logical halves of the work area (POINTL).

The updated checkpoint record is then written back on the checkpoint track.

The output sequence block count (OUTPTG) is calculated by multiplying the input sequence block count (INPUTG) by the order of merge (PH2IOM).

CPOADR-12, CQ-G2

The initial disk addresses for the input and output sequences of the upcoming pass are computed for a maximum order of merge (three or six, as the case may be). The addresses are stored in ARADDR-FRADDR for input, and starting at ORADDR for output. As each initial address is calculated, two values are extracted from the work area table:

1. An index value (multiple of 12), or pointer to the work area table (LIMITS)
2. The logical unit address pertaining to the address

These values are placed adjacent to the disk address in the table.

SHTPTR, CQ-H2

The two hexadecimal pointers to the logical halves of the work area table are reversed. The input area for the current or upcoming pass becomes the output area for the next pass, etc., alternating between the two halves throughout the phase.

NEWITL, CQ-J2

The interleave factors to be used during the pass are calculated ($M/BPT = Q + R$) and stored at IRMDR+1 and IQUOT+1 for input and at ORMDR+1 and OQUOT+1 for output. These factors will be used until it is determined later in the pass that they are to be changed.

NEXTPASS, CC-K2

If the upcoming pass is the last one, the switch at NOTLAS is made a no-op to enable the pass-pass routine to fetch phase 3 into main storage. Until such time, NOTLAS remains a branch to ENDPAS in the merge-merge routine (Chart CP). The number of the upcoming pass (whether for phase 2 or phase 3) is then printed out:

```

      '7DB1I PHASE 2, PASS nn'
or
      '7DC1I PHASE 3, PASS nn'
```

ILEAVE-10, CQ-B3

The number of output sequences to be formed in the upcoming pass is calculated (NSR).

Note: The output sequences in this pass will be the input sequences for the next pass.

INITOM, CQ-C3

The input interleave factors that will be used in the last merge of the pass (when $S \leq OM$) are now calculated and stored in REDUCEI+1 and REDUCEI+5. These are two instructions at label REDUCEI in the merge-merge routine that will place these input interleave factors in IRMDR+1 and IQUOT+1, respectively, when the last merge is entered. The reduced input interleave factors to be used when the order of merge is reduced are stored in IRMDR1+1 and IQUOT1+1.

The merge-merge routine is then initialized with the order of merge to be used during the last merge of the pass. This consists of inserting one of six possible instructions (which are listed beginning at LASTM) at WAY4-4 in the merge-merge routine.

OLEAVE, CQ-E3

The number of merges to be performed during the next pass before the output interleave factors are to be changed is now calculated and stored in MAXFAC. The output interleave factors to be used when MAXFAC is reduced to zero (when $S \leq M^2$) are stored in REDUCEO+1 and REDUCEO+5. These are two instructions at label REDUCEO in the merge-merge routine that will place these output interleave factors in ORMDR+1 and OQUOT+1, respectively, when MAXFAC=0. The reduced output interleave factors to be used when the order of merge is reduced are stored in ORMDR1+1 and OQUOT1+1.

NOTLAS, CQ-F3

As noted in the text under label NEXTPASS, this location is a branch to ENDPAS in the merge-merge routine until it is determined that the upcoming pass will be the last pass. NOTLAS is then made a no-op and the routine assembles and writes the constants for phase 3 on the checkpoint track and fetches phase 3 into main storage.

This routine fills the input areas in main storage with records from the input portion of the disk work area. At the beginning of a merge, all the input areas are filled and the compare loops are initialized. Subsequently, the input areas are refilled individually as they are depleted and, as input sequences are depleted, the compare loops are closed off one by one.

The maximum number of input sequences (A through F for a 6-way merge or A through C for a 3-way merge) are called in. As each block is read into its input area, the disk address of the next block in that sequence is calculated by a separate routine (Chart CY). The interleave factors for calculating the disk addresses are changed when necessary.

The number of sequences that are called into the input areas is determined by the order of merge. When all the main storage input areas are filled, the program continues to the compare loops (Charts CS through CW, as the case may be).

USTOPA, CR-B2

The compare loops are initialized for the required sequences and a corresponding bit is set in the end-of-merge indicator (OMERGE):

<u>Seq.</u>	<u>Label</u>	<u>Initialize Branches at</u>	<u>OMERGE Bit</u>
A	USTOPA	BPUTF, BPUTE, BPUTD, BPUTC, BPUTB, FILLA	1
B	USTOPB	COMPBA, BFA, BEA, BDA, BCA, FILLB	2
C	USTOPC	COMPBC, BFB, BEB, BDB, FILLC	3
D	USTOPD	COMPDC, BFC, BEC, FILLD	4
E	USTOPE	COMPED, BFD, FILLE	5
F	USTOPF	COMPFE, FILLF	6

When all the specified sequences have been processed and the 1-bits in OMERGE have been inverted to 0-bits as described in FILLA, a branch is made to the output routine (Chart CX). Until then, the routine continues to FILLA (or FILLB, etc., as the case may be).

The unconditional branch at this location is a no-op as long as there are records to be processed in the current input sequence; the routine thus continues to GET (or GETB, etc., as the case may be). When the end of the current input sequence is reached, the compare loop for that sequence is closed off and the corresponding bit in OMERGE is inverted to a zero. Then, as long as there are more input sequences, a branch is made to an address that varies according to the current sequence and the order of merge; these addresses are listed in the table on Chart CR.

GETA, CR-E2

This location is entered not only from the preceding function block (FILLA) but also from the various compare loops as long as there are records in the input sequences. As each sequence is depleted, the entry to this routine is at USTOP_n (to close the compare loop for the particular sequence) instead of to this point.

The starting address of the input area for the current sequence (ABEGIN, BBEGIN, etc.) is used along with the corresponding logical unit address to initialize a channel program to get a block of records from disk storage.

RDABCD, CR-F2

An EXCP macro is issued and a block of records is read into the specified input area in main storage.

For all sequences except the highest one (F in a 6-way merge or C in a 3-way merge), the routine continues to LM1 (or LM2, etc., as the case may be). When the highest sequence is being processed, the routine branches directly to IRMDR.

LM1, CR-G3

This location is a switch that will be on (no-op) only during the last merge of a pass. Until such time, the routine branches to IRMDR.

During the last merge of a pass, a test is made to determine if the input interleave factor for the current sequence

should be reduced. This factor is reduced when the number of blocks processed in the current sequence is one greater than the number of blocks in the last sequence of a pass (LSTRGN). The count in LSTRGN is reduced by one each time this function is entered during the last merge of a pass. If the decremented count is equal to or higher than zero, a branch is made to IRMDR; if lower, the branch is to IRMDR1.

IRMDR, CR-H4

This location is entered when the input interleave factors are not to be changed. The factors are stored in RMDR and QUOT and the program continues to the routine to calculate the next interleaved disk address at CALADR (Chart CY). The program returns to this routine at BYPAS1+4.

IRMDR1, CR-H5

This location is entered when the input interleave factors need to be changed. The reduced factors for the next lower order of merge are stored in RMDR and QUOT and a branch is made to the routine to calculate the next interleaved disk address at CALADR (Chart CY). The program returns to this routine at BYPAS1+4.

BYPAS1+4, CR-J3

The current interleaved disk address (CHHR) is stored in ARADDR (or BRADDR, etc., as the case may be) and a branch is made to an address that varies according to the current sequence and the order of merge. These addresses are listed in the table on Chart CR.

SEQUENCE F COMPARE LOOP, VARIABLE-LENGTH RECORDS - CS

The program that was loaded into main storage at the start of phase 2 was for either a 6-way (DSORT204) or a 3-way merge (DSORT201). For a 3-way merge, the compare loops start at sequence C (Chart CV).

For a 6-way merge, the input routine (Chart CR) initialized certain branch instructions in all the compare loops from F through B. However, the flow through these loops varies not only with the order

of merge but also, later on, with the depletion of records in the sequences being merged. As each sequence is depleted, a branch is made to the compare loop for the next lower order of merge (Charts CT through CW, consecutively).

As previously described in the introduction to phase 2 and in the initialization routine (CPIOAS, Chart CN), each input area has an overflow area equal in size to LMAX-1. These overflow areas are used for split records so that when the next block is read in the split record can be compacted and merging can continue. A split record will cause a depleted-input-block condition so that the input area will be refilled and the second part of the split record will thus be available.

A branch-and-link to the output routine (Chart CX) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence F, E, or D. It then exits to the sequence E compare loop (Chart CT).

COMPFE, CS-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence F is depleted. If such is the case, a branch is made to COMPED in the sequence E compare loop (Chart CT); if not, a record from sequence F is compared with a record from sequence E. As long as F is determined to be the winner, it is compared with records from the other available sequences in turn. The routine thus continues in this compare loop or exits to another loop, depending on the results of each comparison. For example, when merging in ascending sequence:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMPFE	F	COMPFD
	E	COMPED (Chart CT)
COMPFD	F	COMPFC
	D	COMPDC (Chart CU)
COMPFC	F	COMPFB
	C	COMPBC (Chart CV)
COMPFB	F	COMPFA
	B	COMPBA (Chart CW)
COMPFA	F	PUTF
	A	PUTA (Chart CW)

The branch exits are determined not only by the results of the comparison but also

by the depletion of input sequences. For example, if in COMPFD sequence D is found to be depleted, the instruction at BFC is an unconditional branch to COMPFC to compare the F record with the next C record.

Another variation in the compare loop operation occurs when, for example, an F record is found to be the winner through COMPFE and COMPFD. Then, in COMPFC, the C record is found to be the winner. The exit from the loop, as previously described, is to COMPCB (Chart CV); however, the return address that is saved in register SAVEC is COMPFC+2. Then, assuming the C record is the winner through compare loops B and A, it is moved to the output area, and control is returned to the F loop at the address in SAVEC. The reason for entering this loop at COMPFC+2 is that although C was the winner, the F record had already been determined to be winner over E and D at that time. Therefore, the comparing in F loop resumes at the point where F is compared with the next C record.

PUTF, CS-H2

The address of the winning record from sequence F is loaded into register MREG and a branch-and-link is made to the OUTFUL in output routine (Chart CX). Control is returned to this routine at NXTFR.

NXTFR, CS-G4

The address for the sequence F input area is updated and a test is made to determine if the end of input sequence F has been reached (hexadecimal F denotes end of sequence). If so, a branch is made to USTOPF in the input routine (Chart CR) to close the F compare loop so that future entries during this merge will branch directly from COMPFE to COMPED. If the F sequence is not yet depleted, one of two courses of action may be taken:

- The next record from sequence F is a split record - branch to SPLITF.
- The next record from sequence F is not a split record - test for sequence F input area depleted.

If the input area is depleted, a branch is made to GETF in the input routine to refill the input area. If the input area is not depleted, a branch is made back to the start of the highest available compare loop (COMPFE) to compare the next record.

SPLITF, CS-J5

The routine initializes to move the first part of the split record to the overflow area adjacent to the F input area. A branch-and-link is then made to SPLITI in the output routine. Upon return to this routine, the real output address (SAVPUT) is reloaded in register PUTOUT and a branch is made to GETF in the input routine. This last branch is made because a split record signals a depleted input area.

SEQUENCE E COMPARE LOOP, VARIABLE-LENGTH RECORDS - CT

For a 5-way merge, the input routine (Chart CR) initialized certain branch instructions in all the compare loops from E through B. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the sequences being merged. As each sequence is depleted, a branch is made to the compare loop for the next lower order of merge (Charts CU through CW, consecutively).

As previously described in the introduction to phase 2 and in the initialization routine (CPIOAS, Chart CN), each input area has an overflow area equal in size to LMAX-1. These overflow areas are used for split records so that when the next block is read in the split record can be compacted and merging can continue. A split record will cause a depleted-input-block condition so that the input area will be refilled and the second part of the split record will thus be available.

A branch-and-link to the output routine (Chart CX) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence E. It then exits to the sequence D compare loop (Chart CU).

COMPED, CT-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence E is depleted. If such is the case, a branch is made to COMPDC in the sequence D compare loop (Chart CU); if not, a record from sequence E is compared with a record from sequence D. As long as E is determined to be the winner, it is compared with records from the other

available sequences in turn. The routine thus continues in this compare loop or exits to another loop, depending on the results of each comparison. For example, when merging in ascending sequence:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMPED	E	COMPEC
	D	COMPDC (Chart CU)
COMPEC	E	COMPEB
	C	COMPCCB (Chart CV)
COMPEB	E	COMPEA
	B	COMPBA (Chart CW)
COMPEA	E	PUTE
	A	PUTA (Chart CW)

The branch locations are determined not only by the results of the comparison but also by the depletion of input sequences. For example, if in COMPEC sequence C is found to be depleted, the instruction at BEB is an unconditional branch to COMPEB to compare the E record with the next B record.

Another variation in the compare loop operation occurs when, for example, an E record is found to be the winner through COMPED and COMPEC. Then, in COMPEB, the B record is found to be the winner. The exit from the loop, as previously described, is to COMPBA (Chart CW); however, the return address that is saved in register SAVEB is COMPEB+2. Then, assuming the B record is the winner through compare loop B, it is moved to the output area, and control is returned to the E loop at the address in SAVEB. The reason for entering this loop at COMPEB+2 is that although B was the winner, the E record had already been determined to be winner over D and C at that time. Therefore, the comparing in E loop resumes at the point where E is compared with the next B record.

PUTE, CT-G2

The address of the winning record from sequence E is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CX). Control is returned to this routine at NXTER.

NXTER, CT-G4

The address for the sequence E input area is updated and a test is made to determine if the end of input sequence E has been

reached (hexadecimal F denotes end of sequence). If so, a branch is made to USTOPE in the input routine (Chart CR) to close the E compare loop so that future entries during this merge will branch directly from COMPED to COMPDC. If the E sequence is not yet depleted, one of two courses of action may be taken:

- The next record from sequence is a split record - branch to SPLITE.
- The next record from sequence E is not a split record - test for sequence E input area depleted.

If the input area is depleted, a branch is made to GETE in the input routine to refill the input area. If the input area is not depleted, a branch is made back to the start of the highest available compare loop (COMPFE) to compare the next record.

SPLITE, CT-J5

The routine initializes to move the first part of the split record to the overflow area adjacent to the E input area. A branch-and-link is then made to SPLITI in the output routine. Upon return to this routine, the real output address (SAVPUT) is reloaded in register PUTOUT and a branch is made to GETE in the input routine. This last branch is made because a split record signals a depleted input area.

SEQUENCE D COMPARE LOOP, VARIABLE-LENGTH RECORDS - CU

For a 4-way merge, the input routine (Chart CR) initialized certain branch instructions in all the compare loops from D through B. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the sequences being merged. As each sequence is depleted, a branch is made to the compare loop for the next lower order of merge (Charts CV and/or CW).

As previously described in the introduction to phase 2 and in the initialization routine (CPIOAS, Chart CN), each input area has an overflow area equal in size to LMAX-1. These overflow areas are used for split records so that when the next block is read in the split record can be compacted and merging can continue. A split record will cause a depleted-input-block condition so that the input area will be refilled and the second part of the split record will thus be available.

A branch-and-link to the output routine (Chart CX) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence D. It then exits to the sequence C compare loop (Chart CV).

COMPDC, CU-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence D is depleted. If such is the case, a branch is made to COMPDB in the sequence C compare loop (Chart CV); if not, a record from sequence D is compared with a record from sequence C. As long as D is determined to be the winner, it is compared with records from the other available sequences in turn. The routine thus continues in this compare loop or exits to another loop, depending on the results of each comparison. For example, when merging in ascending sequence:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMPDC	D	COMPDB
	C	COMPDB (Chart CV)
COMPDB	D	COMPDA
	B	COMPBA (Chart CW)
COMPDA	D	PUTD
	A	PUTA (Chart CW)

The branch locations are determined not only by the results of the comparison but also by the depletion of input sequences. For example, if in COMPDB sequence B is found to be depleted, the instruction at BDA is an unconditional branch to COMPDA to compare the D record with the next A record.

Another variation in the compare loop operation occurs when, for example, a D record is found to be the winner in COMPDC. Then, in COMPDB, the B record is found to be the winner. The exit from the loop, as previously described, is to COMPBA (Chart CW); however, the return address that is saved in register SAVEB is COMPDB+2. Then, assuming the B record is the winner in compare loop A, it is moved to the output area, and control is returned to the D loop at the address in SAVEB. The reason for entering this loop at COMPDB+2 is that although B was the winner, the D record had already been determined to be winner over C at that time. Therefore, the comparing in D loop resumes at the point where D is compared with the next B record.

PUTD, CU-F2

The address of the winning record from sequence D is loaded into register MREG and a branch-and-link is made to OUTFUL of the output routine (Chart CX). Control is returned to this routine at NXTDR.

NXTDR, CU-G4

The address for the sequence D input area is updated and a test is made to determine if the end of input sequence D has been reached (hexadecimal F denotes end of sequence). If so, a branch is made to USTOPD in the input routine (Chart CR) to close the D compare loop so that future entries during this merge will branch directly from COMPDC to COMPDB. If the D sequence is not yet depleted, one of two courses of action may be taken:

- The next record from sequence D is a split record - branch to SPLITD.
- The next record from sequence D is not a split record - test for sequence D input area depleted.

If the input area is depleted, a branch is made to GETD in the input routine to refill the input area. If the input area is not depleted, a branch is made back to the start of the highest available compare loop (COMPFE) to compare the next record.

SPLITD, CU-J5

The routine initializes to move the first part of the split record to the overflow area adjacent to the D input area. A branch-and-link is then made to SPLITI in the output routine. Upon return to this routine, the real output address (SAVPUT) is reloaded in register PUTOUT and a branch is made to GETD in the input routine. This last branch is made because a split record signals a depleted input area.

SEQUENCE C COMPARE LOOP, VARIABLE-LENGTH RECORDS - CV

This compare loop is entered:

- In a 6-way merge, when the record from sequence C is found to be the winner in a previous compare loop.

- In a 6-way merge, when sequences F, E, and D have been depleted.
- In a 3-way merge, at the completion of the input routine.

For a 3-way merge, the input routine (Chart CR) initialized certain branch instructions in compare loops C and B. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the sequences being merged. As each sequence is depleted, a branch is made to the compare loop for the next lower order of merge (Chart CW).

As previously described in the introduction to phase 2 and in the initialization routine (CPIOAS, Chart CN), each input area has an overflow area equal in size to LMAX-1. These overflow areas are used for split records so that when the next block is read in the split record can be compacted and merging can continue. A split record will cause a depleted-input-block condition so that the input area will be refilled and the second part of the split record will thus be available.

A branch-and-link to the output routine (Chart CX) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence C. It then exits to the sequence B compare loop (Chart CW).

COMPCB, CV-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence C is depleted. If such is the case, a branch is made to COMPBA in the sequence B compare loop (Chart CW); if not, a record from sequence C is compared with a record from sequence B. If the C record is the winner, it is compared with a record from sequence A. If the C record wins again, the routine continues to PUTC.

The other branch locations, in the event that either B or A is determined to be the winner, are:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMPCB	B	COMPBA (Chart CW)
COMPCA	A	PUTA (Chart CW)

The branch locations are determined not only by the results of the comparison but also by the depletion of input sequences.

For example, if in COMPCA sequence A is found to be depleted, the instruction at BPUTC is an unconditional branch to PUTC to prepare to move the C record to the output area.

Another variation in the compare loop occurs when, for example, a C record is found to be the winner in COMPCB. Then, in COMPCA, the A record is found to be the winner. The exit from the loop, as previously described, is to PUTA (Chart CW); however, the return address that is saved in register SAVEA is COMPCA+2. Then, after the A record is moved to the output area, control is returned to the C loop at the address in SAVEA. The reason for entering this loop at COMPCA+2 is that although A was the winner, the C record had already been determined to be winner over the B record. Therefore, the comparing in C loop resumes at the point where C is compared with the next A record.

PUTC, CV-E2

The address of the winning record from sequence C is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CX). Control is returned to this routine at NXTCR.

NXTCR, CV-G3

The address for the sequence C input area is updated and a test is made to determine if the end of input sequence C has been reached (hexadecimal F denotes end of sequence). If so, a branch is made to USTOPC in the input routine (Chart CR) to close the C compare loop so that future entries during this merge will branch directly from COMPCB to COMPBA. If the C sequence is not yet depleted, one of two courses of action may be taken:

- The next record from sequence C is a split record - branch to SPLITC.
- The next record from sequence C is not a split record - test for sequence C input area depleted.

If the input area is depleted, a branch is made to GETC in the input routine to refill the input area. If the input area is not depleted, a branch is made back to the start of the highest available compare loop (depending on the order of merge) to compare the next record.

SPLITC, CV-J4

The routine initializes to move the first part of the split record to the overflow area adjacent to the C input area. A branch-and-link is then made to SPLITI in the output routine. Upon return to this routine, the real output address (SAVPUT) is reloaded in register PUTOUT and a branch is made to GETC in the input routine. This last branch is made because a split record signals a depleted input area.

SEQUENCE B COMPARE LOOP, VARIABLE-LENGTH RECORDS - CW

For a 2-way merge, the input routine (Chart CR) initialized certain branch instructions in compare loop B. However, the flow through this loop varies not only with the order of merge but also, later on, with the depletion of records in sequence B. When sequence B is depleted, a branch is made to prepare to move the A record to the output area.

As previously described in the introduction to phase 2 and in the initialization routine (CPIOAS, Chart CN), each input area has an overflow area equal in size to LMAX-1. These overflow areas are used for split records so that when the next block is read in the split record can be compacted and merging can continue. A split record will cause a depleted-input-block condition so that the input area will be refilled and the second part of the split record will thus be available.

A branch-and-link to the output routine (Chart CX) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from sequence B. It then branches directly to PUTA.

COMPBA, CW-B2

The exit from this loop is provided at the beginning so that no processing need be done when sequence B is depleted. If such is the case, a branch is made to PUTA; if not, a record from sequence B is compared with a record from sequence A. If the B record is the winner, the routine branches to PUTB; if the A record is the winner, the routine branches to PUTA.

PUTB, CW-D2

The address of the winning record from sequence B is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CX). Control is returned to this routine at NXTBR.

NXTBR, CW-G2

The address for the sequence B input area is updated and a test is made to determine if the end of input sequence B has been reached (hexadecimal F denotes end of sequence). If so, a branch is made to USTOPB in the input routine (Chart CR) to close the B compare loop so that future entries during this merge will branch directly from COMPBA to PUTA. If the sequence is not yet depleted, one of two courses of action may be taken:

- The next record from sequence B is a split record - branch to SPLITB.
- The next record from sequence B is not a split record - test for sequence B input area depleted.

If the input area is depleted, a branch is made to GETB in the input routine to refill the input area. If the input area is not depleted, a branch is made back to the start of the highest available compare loop (depending on the order of merge) to compare the next record.

SPLITB, CW-J3

The routine initializes to move the first part of the split record to the overflow area adjacent to the B input area. A branch-and-link is then made to SPLITI in the output routine. Upon return to this routine, the real output address (SAVPUT) is reloaded in register PUTOUT and a branch is made to GETB in the input routine. This last branch is made because a split record signals a depleted input area.

PUTA, CW-D4

The address of the winning record from sequence A is loaded into register MREG and a branch-and-link is made to OUTFUL in the output routine (Chart CX). Control is returned to this routine at NXTAR.

NXTAR, CW-G4

The address for the sequence A input area is updated and a test is made to determine if the end of input sequence A has been reached (hexadecimal F denotes end of sequence). If so, a branch is made to USTOPA in the input routine (Chart CR) to close the sequence A compares. If the A sequence is not yet depleted, one of two courses of action may be taken:

- The next record from sequence A is a split record - branch to SPLITA.
- The next record from sequence A is not a split record - test for sequence A input area depleted.

If the input area is depleted, a branch is made to GETA in the input routine to refill the input area. If the input area is not depleted, a branch is made back to the start of the highest available compare loop (depending on the order of merge) to compare the next record.

SPLITA, CW-J5

The routine initializes to move the first part of the split record to the overflow area adjacent to the A input area. A branch-and-link is then made to SPLITI in the output routine. Upon return to this routine, the real output address (SAVPUT) is reloaded in register PUTOUT and a branch is made to GETA in the input routine. This last branch is made because a split record signals a depleted input area.

OUTPUT ROUTINE, VARIABLE-LENGTH RECORDS - CX

This routine is entered at one of four locations:

- OUTFUL, when a winning record has been found in one of the compare loops.
- MMPP, when an end-of-merge condition is detected in the input routine (end of all input sequences).
- SPLITI, when a split record has been found in one of the input areas during the compare process.
- SPLITM, when the second part of a split record is to be moved to the output area.

When the routine is entered at OUTFUL, and the entire winning record will fit in the output area, the winning record is moved to the output area. The program then branches back to NXTAR (or NXTBR, etc.) in the compare loop in which the winning record was found. When the winning record will not fit in the output area, a split output record is indicated and the portion that will fit is moved to the output area. The output area is then written on disk and the second part of the winning record is moved to the start of the output area.

When the routine is entered at MMPP, an end-of-sequence indicator (hexadecimal F0) is inserted in the last byte of the current output block and the end-of-merge switch is turned on. The last output block is then written out, the output interleave factor is changed, if necessary, and the new disk address is calculated (Chart CY). The program then branches to the merge-merge routine (Chart CP).

When the routine is entered at SPLITI, the first portion of the split record is moved to the overflow area adjacent to the input area for the sequence. The program then branches back to GETA (or GETB, etc.) in the input routine (Chart CR).

When the routine is entered at SPLITM, the second part of the winning record that could not fit in the output area (when the routine was originally entered at OUTFUL) is now moved to the output area.

When the output area is full after a move, the contents are first written in the output portion of the disk work area, the input interleave factor is reduced, if necessary, and the new disk address is calculated (Chart CY). The winning record is then moved to the output area and the program returns to the compare loop in which the winning record was found.

OUTFUL, CX-B2

The record-length indicator of the record to be moved is extracted and used to determine if the entire record will fit in the available output area. If so, the routine continues to SPLITM; if the record is too long to fit, a branch is made to SPLITO.

SPLITI, CX-C1

The split-record indicator (hexadecimal C) is erased and the real output address is saved (to be restored on return to the compare loops). The number of bytes to be moved to the input overflow area are then calculated and a branch is made to SPLITM.

SPLITO, CX-B3

The number of bytes in the second part of the winning record (the part that will not fit in the output area) is calculated. Then the number of bytes in the first part (the part that will fit) is calculated. The split-record indicator is inserted in the first part of the record and the switch at OSPLIT (Chart CY) is turned on so that the second part of the record will be moved after the output block has been written on disk.

SPLITM, CX-D2

Registers are initialized to move a record or a part of a record and a branch is made to VARMOV.

VARMOV, CX-E2

One of three types of moves is performed:

- An entire winning record to the output area
- The first or second part of a winning record to the output area
- The first part of a split input record to the corresponding input overflow area.

If the number of bytes to be moved is 256 or less, the exact number of bytes is moved to the output area or to the input overflow area, as the case may be. For longer moves, a 256-byte portion is moved and registers are adjusted for the length of the remaining portion and for the new output area address. This operation is repeated until the remaining portion of the record is 256 bytes or less in length. The length of the remaining portion is then calculated and the final move is executed.

After the required number of bytes has been moved, the next available output area

address is calculated for the next move and for testing if the output area is full.

The current address of the output area (in register PUTOUT) is compared to the address of the end of the output area (OUTEND). If the output area is full, a branch is made to WRITE; if not, a branch is made to NXTAR (or NXTBR, etc.) in the compare loop in which the winning record was found or, in the case of split input records, to GETA (or GETB, etc.) in the input routine. Note that when the first part of a split output record has been moved to the output area, the output area will be full.

MMPP, CX-B4

The end-of-sequence indicator (hexadecimal F0) is inserted at the end of the block currently in the output area and the end-of-merge switch (MMPP1 on Chart CY) is turned on. The routine then continues to WRITE.

WRITE, CX-C4

The address of the start of the output area (OBEGIN) is restored in register PUTOUT, and the current disk address of the output sequence is moved to the current disk interleave address. If the BPT (blocks per track) is greater than 1, a branch is made to WTDATA; if BPT = 1 (a condition that can occur only during pass 1), the output count field address is stored, the current interleave address is reduced by one, and the output area address in register PUTOUT is incremented by eight to put it past the count field. The routine then continues to WTDATA.

WTDATA, CX-D4

An EXCP macro is issued and the block of records is written from the main storage output area into the output portion of the disk work area. The parameters for the write operation are supplied by the command control block at OCCB. The routine then continues to LMO.

LMO, CX-E4

This location is a switch that will be on (no-op) only when the remaining input sequences are equal to or less than the order of merge squared ($S \leq M^2$). This condition is detected in the merge-merge routine (see LMOSW, Chart CP-F3). Until such time, LMO is a branch to ORMDR.

When $S \leq M^2$, is a calculation and test are made to determine if the number of blocks merged to the output sequence is greater by one than the number of blocks contained in the last merge of the pass. If it is, the output interleave factors are to be reduced and a branch is made to ORMDR1; otherwise, the routine continues to ORMDR.

ORMDR, CX-G4

This location is entered when the output interleave factors are not to be changed. The factors are stored in RMDR and QUOT and the program branches to the routine to calculate the next interleaved disk address (Chart CY).

ORMDR1, CX-G5

This location is entered when the output interleave factors need to be changed. The reduced factors for the next lower order of merge are stored in RMDR and QUOT and the program branches to the routine to calculate the next interleaved disk address (Chart CY).

CALCULATE INTERLEAVED DISK ADDRESS ROUTINE,
VARIABLE-LENGTH RECORDS - CY

This routine is entered from the input routine (Chart CR) whenever a block is read, or from the output routine (Chart CX) whenever a block is written.

The current interleaved disk address is updated with the factors in RMDR and QUOT, which have been placed there by the routine (input or output) immediately before entry to this routine.

The record number and head/track number are calculated and checked for validity. If maximums are exceeded, the next higher valid address is calculated. The upper limit of the current work area extent is then checked and, if exceeded, a new

address is calculated based on the lower limit of the next logical segment of the work area.

See Figure 37 for an illustration of interleaved disk address calculations.

CALADR, CY-C2

The interleave factor in RMDR is used to calculate the next record number by adding it to the current disk address (CHHR) in register 0. If the new record number is greater than the maximum number of sort blocks per track (BPT), it is not valid; the next valid record number is then calculated by adding the 256-complement of the BPT (BCOMP) to the value in register 0.

The interleave factor in QUOT is then used to calculate the new head/track number by adding it to the newly-calculated record address in register 0. If the new head/track number exceeds nine, it is not valid; the next valid cylinder/head number is calculated by adding the 256-complement of ten (HCOMP) to the value in register 0.

A test is then made to determine if the upper limit of the current work area extent has been exceeded by the new disk address just calculated in register 0. If not, the new address is valid and a branch is made to LMTSOK; if so, the disk address is re-calculated based on the lower limit of the next extent in the work area.

LMTSOK, CY-E3

The new interleaved disk address, for input or output, as the case may be, is stored in CHHR. The sequence block count (BLOCKC) is reduced by one.

BSTR6, CY-F3

If the interleaved address just calculated and stored was for output, it is moved from CHHR to ORADDR and the routine continues to OSPLIT. If the address was for input, a branch is made to BYPAS1+4 (or BYPAS2+4, etc., as the case may be) in the input routine (Chart CR), where the address will be moved from CHHR to ARADDR (or BRADDR, etc., as the case may be).

OSPLIT, CY-H3

This location is normally a branch to MMPP1, the end-of-merge switch. However, when a winning record is split because it would not fit in the output area, OSPLIT would have been made a no-op (at SPLITO in the output routine). The routine now initializes to move the second part of the split output record and branches back to SPLITM in the output routine (Chart CX).

MMPP1, CY-H4

This location is normally a branch back to BYPAS1+4 (or BYPAS2+4, etc.) in the input routine (Chart CR). At the end of a merge, however, MMPP1 would have been changed to a no-op (at MMPP in the output routine) so that the program will continue to MMPP2 in the merge-merge routine (Chart CP).

Order of Merge -- 4
 Blocks Per Track -- 5

RMDR -- 0004
 QUOT -- 0000
 BCOMP -- 00FB

Current Disk Address	C	H	H	R	
	01	00	08	01	Block 1
Add (RMDR) Factor			+	00 04	
Record Number Valid (Equal to or Less Than BPT)	01	00	08	05	
Add (Quot) Factor			+	00 00	
Track Number Valid (Equal to or Less Than 09)	01	00	08	05	Block 2

Current Disk Address	C	H	H	R	
	01	00	08	05	Block 2
Add (RMDR) Factor			+	00 04	
Record Number Invalid	01	00	08	09	
Add Complement of BPT (05=00FB)			+	00 FB	
Record Number Valid	01	00	09	04	
Add (Quot) Factor			+	00 00	
Track Number Valid	01	00	09	04	Block 3

Current Disk Address	C	H	H	R	
	01	00	09	04	Block 3
Add (RMDR) Factor			+	00 04	
Record Number Invalid	01	00	09	08	
Add Complement of BPT			+	00 FB	
Record Number Valid	01	00	0A	03	
Add (Quot) Factor			+	00 00	
Track Number Invalid	01	00	0A	03	
Add Complement of 10	00	FF	F6	00	
	02	00	00	03	Block 4

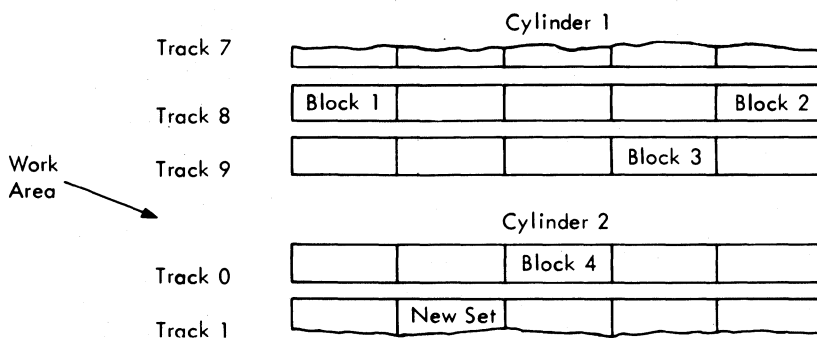


Figure 37. Calculate Interleaved Disk Address

Phase 3 consists of one of four overlays, depending on the order of merge and the input record type:

- DSORT301 (4-way merge, fixed-length), for an order of merge from 1 to 4 and either a fixed-length record sort or the ADDRROUT option per fixed- or variable-length records.
- DSORT302 (7-way merge, fixed-length), for an order of merge from 5 to 7 and either a fixed-length record sort or the ADDRROUT option for fixed- or variable-length records.
- DSORT303 (3-way merge, variable-length), for an order of merge from 1 to 3 and variable-length records.
- DSORT304 (6-way merge, variable-length), for an order of merge from 4 to 6 and variable-length records.

The overlay to be used in this phase was determined by phase 1 and called in by phase 2.

For the purpose of describing the program logic, this phase has been divided into two general categories:

- Fixed-length records (7-way and 4-way merges), Charts DA through DL
- Variable-length records (6-way and 3-way merges), Charts DM through DW

This introduction serves for both categories. Where necessary, duplicate figures and charts are provided (with the required differences, if any) so that each category is complete in itself. For example, there are two major-component charts (04).

Phase 3 is initialized for either disk or tape output. Routines for output and label linkage are relocated for tape output.

Phase 3 performs the final merge pass, creating the output file from the sequences located in the input half of the disk work

area. The number of input sequences is equal to or less than the order of merge used in phase 2. Merging is accomplished in a manner similar to the last merge of a phase 2 pass except that the output is written consecutively.

The read-input-data and the mainline compare routines are similar to phase 2. The compute-interleave-address routine is also similar to its phase 2 counterpart, except that the output from the sort is not interleaved.

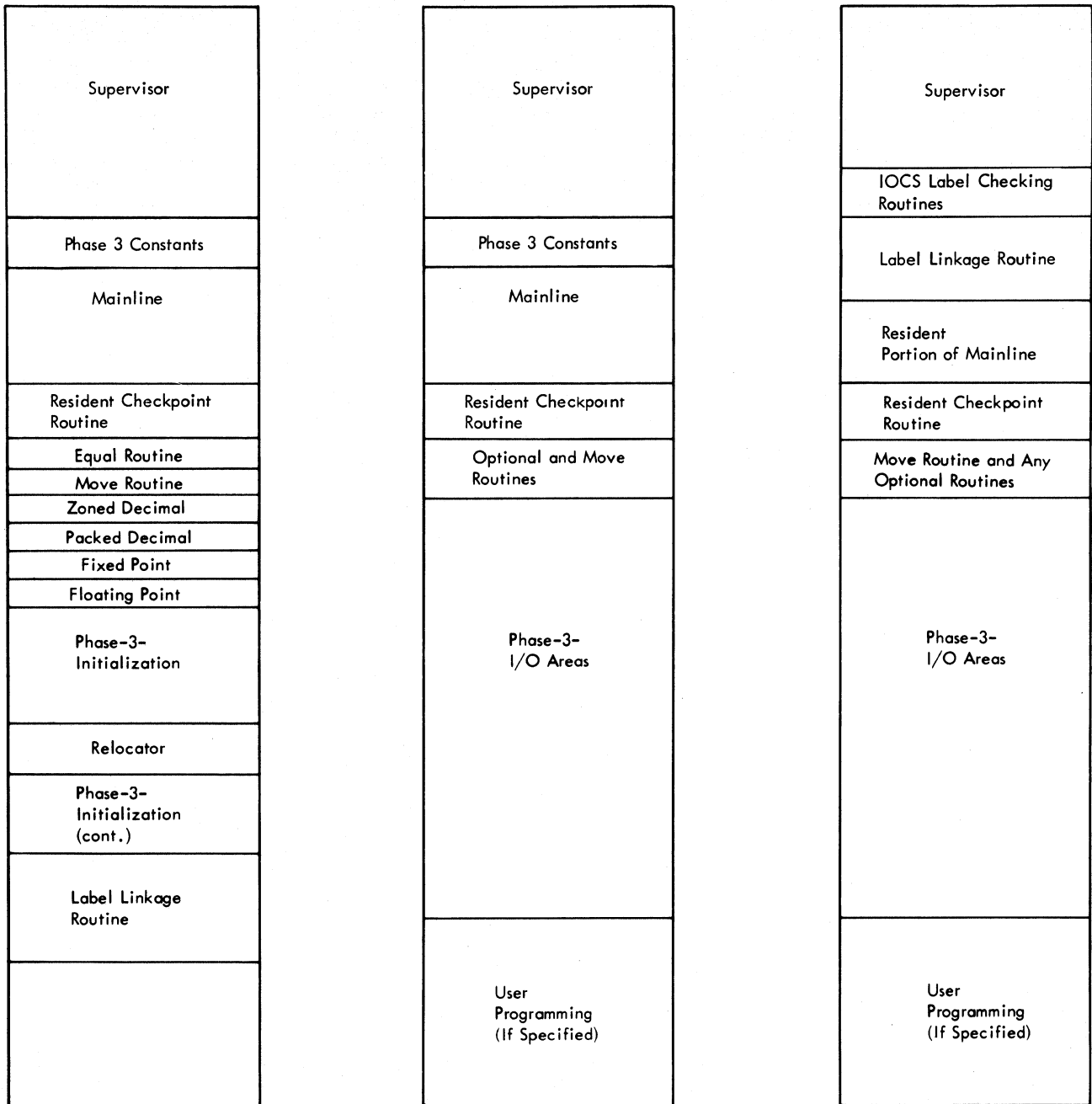
After a winning record has been determined by the compare routine, it is sequence-checked with the previous record moved to the output area. Following the sequence check, the previous record is converted in the output area if data conversion has been specified by the user.

The label-linkage routine is initialized for label processing of the output file. Linkage to the transient IOCS label-checking routine is initialized within this routine, which is entered for open, close, end of volume, sequence error, and end of job.

User programming, in phase 3, is accessed via exit 31 and exit 32. Exit 31, located within the label-linkage routine, permits the user to create and write non-standard header and trailer labels for tape volumes, or to create user header and trailer labels for disk or tape volumes. Exit 32, located in the output routine, is available after each logical record has been moved into the main storage output area in its proper sequence and format, as determined by the user SORT control card.

When the last logical record has been merged and written into the output file, the label-linkage routine passes control to IOCS to close the output file. The end-of-job routine, indicating completion of the sort operation, is entered when IOCS returns control.

Figure 38 illustrates main storage layout for phase 3 for fixed-length records. For variable-length records, the layout is identical except that the move routine is deleted.



Start of Phase-3-

End of Phase - 3-
Initialization or
During
Merging in Phase 3

"Open", "EOV", "Close",
and "Sequence Error"
Times or at End-of-
Job

Note: For variable-length records, the core layout is identical except that no reference is made to the move-routine.

Note: Not drawn to scale.

Figure 38. Phase 3 Main Storage Layout

PHASE 3 INITIALIZATION, FIXED-LENGTH
RECORDS - DA

The checkpoint record and the phase 2 constants pertinent to phase 3 are read into main storage and several routines are initialized:

- Mainline compare routine.
- ADDRROUT and exit 32 routines.
- Disk or tape output routine.
- Disk or tape label-linkage routine.
- Relocatable routines.

The interleave factors used for the disk input routine are calculated, as are the constants for phase 3 input/output areas.

For disk output, the output routine is executed as shown in the listing. For tape output, the routines at labels OP1EOV and EOJTAP are relocated to LWRITE+4 and PH3EOJ, respectively, at initialization time.

A check is made to determine if user programming is to be included. Exits available to the user in this phase are 31 and 32.

After initialization is complete, control is passed to the input-data routine.

INTPH3, DA-B1

Job control, employing the system loader, loads phase 3 into main storage following the supervisor. The transfer address is identified by the label INTPH3. Base register 11 is loaded, and registers 2 and 3 are stored. These two registers contain the logical unit address and the disk address (CHHR) of the checkpoint record, respectively. The checkpoint record, created by the assignment phase and updated by phases 1 and 2, is read into main storage (starting at CKPTRD) by the channel program.

PH3CON, DA-C1

The channel program is modified to read phase 3 constants from the checkpoint track into main storage, starting at IPTCCB and continuing through GRADDR for a 7-way merge or DRADDR for a 4-way merge. A

branch-and-link is made to CHEKPT to read these constants into main storage and control is returned at TESTEQ.

TESTEQ, DA-D1

If multiple control data fields are specified in the SORT control card, the equal routine is required and initialization of the mainline compare branches is bypassed. If the equal routine is not required, the branch instructions following each mainline compare are initialized accordingly. For example, if a record from sequence 4 is equal to a record from sequence 3, the next compare is between records from sequences 3 and 2, not sequences 4 and 2.

The sequence error routine is initialized according to the unit assignments. If SYSLOG is a 1052 Printer-Keyboard, the operator may either ignore a sequence error and continue processing or cancel the job when a sequence error is detected. If SYSLOG is not a 1052 Printer-Keyboard, a sequence error automatically cancels the job.

ITCOMP, DA-E1

The mainline compare loops are initialized for length, location, and collating sequence of control data field 1. This information is contained in the 96-byte table (CF1LCT) in the checkpoint record.

The mainline is initialized to update:

- each input area with the length of user record after a winning record has been moved to the output area (as in phase 2). When the ADDRROUT option is being performed, the length of the record is CF+10.
- the output area with the length of the output record after a winning record has been moved to the output area. This length may be equal to, less than, or greater than the length of the input logical record.

OPTION, DA-H1

A test is made to determine if the ADDRROUT option has been specified. If it has, and the control field data is not part of the disk address, a switch is set at NOCONV.

This prevents any translation of the control data associated with the 10-byte tag.

A test is made to determine if exit 32 is specified and, if it is, the switch at NOCONV is bypassed. If exit 32 is not specified, the output routine is initialized to bypass the exit.

RECORD0, DA-J1

The number of records processed by phase 1 (RECPH1) is obtained from the checkpoint record. If the number of records processed is greater than zero, a branch is made to INTLEAVE; if zero, the branch is to PRTEOJ (Chart DP).

INTLEAVE, DA-B3

The interleave factors are calculated and the input disk address routine is initialized:

The order of merge to be used in phase 3 (number of remaining sequences) is divided by BPT (number of sort blocks per 2311 track). The quotient and remainder are the initial interleave factors for accessing the input to phase 3. The OM is reduced by one and the process is repeated to obtain the reduced interleave factors. If OM is equal to one, reduced factors are not calculated.

The input disk address routine is initialized by storing the interleave factors in LRMDR+1 and LQUOT+1, and the reduced interleave factors in LRMDRI+1 and LQUOTI+1.

TAPLLR, DA-C3

For disk output, this function is bypassed by branching to OUTDSK. For tape output, the CCB (OCCB) and the CCW (WTCCW) are initialized to write tape. The routines for OPEN and EOJ (OP1EOV), and the routine to write the last output tape block and to link to close file (EOJTAP), are relocated and initialized.

The LLR (label-linkage routine) is initialized to set the rewind code for close time in the DTF table immediately after the OPEN has been executed. This rewind code is in effect at end-of-volume time (multi-volume file). For instance, if

the user specifies UNLD (unload), each volume is rewound and unloaded at EOJ time, and the next volume is rewound (see Figure 39).

If nonstandard labels or no labels are specified, the tape mark option is initialized. This applies only to an OPEN condition. In this case, the user has the option of inserting or not inserting a tape mark prior to the first record of each output volume. In the case of standard labels, a tape mark is always written by IOCS.

If non-standard or additional user header and trailer labels have been specified, the DTF is initialized to enable IOCS to link to the LLR which, in turn, links to the user through exit 31 to process labels.

At this point, the initialization steps for tape output only are complete and a branch is made to CPLINK to continue initializing for disk and tape.

NO ALTERNATE DRIVE Remarks	USER'S REWIND SPECIFICATIONS		ALTERNATE DRIVE/S Remarks
	OPEN	CLOSE	
The volume is rewound and unloaded. The next volume is rewound.	RWD	RWD	The volume is rewound. The next volume is rewound.
The volume is rewound and unloaded. The next volume is rewound.	RWD	UNLD	The volume is rewound and unloaded. The next volume is rewound.
The volume is rewound and unloaded. The next volume is not rewound.	RWD	NORWD	The volume is not rewound, and the next volume is not rewound.
The volume is rewound and unloaded. The next volume is rewound.	NORWD	RWD	The volume is rewound, and the next volume is rewound.
The volume is rewound and unloaded. The next volume is rewound.	NORWD	UNLD	The volume is rewound and unloaded. The next volume is rewound.
The volume is rewound and unloaded. The next volume is not rewound.	NORWD	NORWD	The volume is not rewound, and the next volume is not rewound.

Figure 39. Rewind Action Taken at End-of-Volume Time for Multi-Volume Tape Files

OUTDSK, DA-D4

Portions of Phase 3 are initialized for disk output:

- Output disk address routine with the

number of blocks that will be written on each output track (BPTO).

- Output channel program to verify each output block as it is written on disk, if VERIFY option is specified.
- Disk DTFSD for additional user labels, if exit 31 is specified.
- Label-linkage routine, by overlaying DTFSD (disk) onto DTFMT (tape).

CPLINK, DA-F3

The label-linkage routine is written on the checkpoint track.

Whenever an OPEN, EOV (tape only), CLOSE, or sequence error condition exists, the label-linkage routine is read into main storage and the condition is processed.

RTNEQ, DA-G3

The relocator routine is initialized to include the equal routine (if number of control data fields is greater than one) and the move routine.

Note: The move routine is not optional but its size is.

If user's format does not require conversion, phase 3 is initialized to bypass the linkage to the conversion routine.

A branch-and-link is then made to the relocator routine (Chart FA) to initialize and relocate the move, reconversion, and equal routines. Control is returned by the relocator at START.

START, DA-J3

The constants required for the remaining initialization routines are obtained from the checkpoint record and stored at CTDLDL - PHEX34. These constants contain output data length, output block length, and information pertinent to user-programmed routines in phase 3.

The relocator routine has stored, in the full word constant RLISA, the main storage address of the first available byte to be used by the phase 3 input/output areas. The remaining initialization routines are

relocated, starting at the main storage address contained in RLISA. This is necessary because, in the case of disk output, initialization of the output count field (key length and data length) could possibly destroy unexecuted instructions in routines starting at OUTAPE and ending at PHEX34.

START1, DA-B5

The input areas are allocated for reading the input sequences into main storage. The number of input areas required is equal to the number of sequences to be merged (PH3IOM) in phase 3. Each input area is equal in length to the sort block size (SORTL).

The starting and ending addresses for each input area are calculated and stored in the constants AEND through GBEGIN (DBEGIN for 4-way merge). For example, AEND contains the end address (address of last byte) of the input sequence 1 block (sequence A from phase 2), and ABEGIN contains the starting address of the input sequence 1 block. If disk output has been specified, the 8-byte count field (the field immediately following the last input area and adjacent to the first byte of the output area) is initialized. The count field becomes part of each output block and is written out with the data portion and key portion (if specified). The output CCW is initialized with the actual data count for either disk or tape output.

OUTAPE, DA-C5

The output area constants OBEGIN-OUTEND are calculated, and the data count for the input CCW is initialized with the sort block length.

The size of the output area depends upon the user specification in the SORT control statements.

PH3MRG, DA-D5

Phase 3 is initialized to execute during the final pass:

- a 1,2,3,4,5,6, or 7-way merge (for 7-way merge)
- a 1,2,3, or 4-way merge (for 4-way merge)

address contained in ABEGIN, and a sequence 1 block is read into the input area.

If the interleave factors need not be reduced, a branch is made to LRMDR (Chart DH) to calculate the next input interleave address; if the interleave factors are to be reduced, the next input address is calculated at LRMDR1 (Chart DH). Upon return to this routine, the calculated address is stored in ARADDR.

For more than a 1-way merge, a branch is made to USTOP2; for a 1-way merge, the branch is to PUT1 (Chart DG).

At the end of sequence 1, this routine is entered at USTOP1 to close the mainline compares for this sequence. A branch is then made to the location stored in register SAVEA, depending on the current sequence and the order of merge. These locations are listed in the tables on Chart DB.

USTOP2, DB-B2

The mainline compares are initialized to open sequence 2, channel programs are prepared with the main storage input area address contained in BBEGIN, and a sequence 2 block is read into the input area.

If the interleave factors need not be reduced, a branch is made to LRMDR (Chart DH) to calculate the next input interleave address; if the interleave factors are to be reduced, the next input address is calculated at LRMDR1 (Chart DH). Upon return to this routine, the calculated address is stored in BRADDR.

For more than a 2-way merge, a branch is made to USTOP3; for a 2-way merge, the branch is to COMP21 (Chart DG).

At the end of sequence 2, this routine is entered at USTOP2 to close the mainline compares for this sequence. A branch is then made to the location stored in register SAVEB, depending on the current sequence and the order of merge. These locations are listed in the tables on Chart DB.

USTOP3, DB-B2

The mainline compares are initialized to open sequence 3, channel programs are prepared with the main storage input area address contained in CBEGIN, and a sequence 3 block is read into the input area.

If the interleave factors need not be reduced, a branch is made to LRMDR (Chart DH) to calculate the next input interleave address; if the interleave factors are to be reduced, the next input address is calculated at LRMDR1 (Chart DH). Upon return to this routine, the calculated address is stored in CRADDR.

For more than a 3-way merge, a branch is made to USTOP4; for a 3-way merge, the branch is to COMP32 (Chart DG) for a 7-way program or to COMP43 (Chart DF) for a 4-way program.

At the end of sequence 3, this routine is entered at USTOP3 to close the mainline compares for this sequence. A branch is then made to the location stored in register SAVEC, depending on the current sequence and the order of merge. These locations are listed in the tables on Chart DB.

USTOP4, DB-B2

The mainline compares are initialized to open sequence 4, channel programs are prepared with the main storage input area address contained in DBEGIN, and a sequence 4 block is read into the input area.

If the interleave factors need not be reduced, a branch is made to LRMDR (Chart DH) to calculate the next input interleave address; if the interleave factors are to be reduced, the next input address is calculated at LRMDR1 (Chart DH). Upon return to this routine, the calculated address is stored in DRADDR.

For more than a 4-way merge, a branch is made to USTOP5; for a 4-way merge, the branch is to COMP43 (Chart DF).

At the end of sequence 4, this routine is entered at USTOP4 to close the mainline compares for this sequence. A branch is then made to the location stored in register SAVED, depending on the current sequence and the order of merge. These locations are listed in the tables on Chart DB.

USTOP5, DB-B2

The mainline compares are initialized to open sequence 5, channel programs are prepared with the main storage input area address contained in EBEGIN, and a sequence 5 block is read into the input area.

If the interleave factors need not be reduced, a branch is made to LRMDR (Chart DH) to calculate the next input interleave address; if the interleave factors are to be reduced, the next input address is calculated at LRMDR1 (Chart DH). Upon return to this routine, the calculated address is stored in ERADDR.

For more than a 5-way merge, a branch is made to USTOP6; for a 5-way merge, the branch is to COMP76 (Chart DC).

At the end of sequence 5, this routine is entered at USTOP5 to close the mainline compares for this sequence. A branch is then made to COMP76 (Chart DC).

USTOP6, DB-B2

The mainline compares are initialized to open sequence 6, channel programs are prepared with the main storage input area address contained in FBEGIN, and a sequence 6 block is read into the input area.

If the interleave factors need not be reduced, a branch is made to LRMDR (Chart DH) to calculate the next input interleave address; if the interleave factors are to be reduced, the next input address is calculated at LRMDR1 (Chart DH). Upon return to this routine, the calculated address is stored in FRADDR.

For more than a 6-way merge, a branch is made to USTOP7; for a 6-way merge, the branch is to COMP76 (Chart DC).

At the end of sequence 6, this routine is entered at USTOP6 to close the mainline compares for this sequence. A branch is then made to COMP76 (Chart DC).

USTOP7, DB-B2

The mainline compares are initialized to open sequence 7, channel programs are prepared with the main storage input area address contained in GBEGIN, and a sequence 7 block is read into the input area.

A branch is made to LRMDR (Chart DH) to calculate the next input interleave address. Upon return to this routine, the calculated address is stored in GRADDR and a branch is made to COMP76 (Chart DC).

At the end of sequence 7, this routine is entered at USTOP7 to close the mainline compares for this sequence. A branch is then made to COMP76 (Chart DC).

MAINLINE COMPARE ROUTINE, FIXED-LENGTH RECORDS - DC, DD, DE, DF, DG

The mainline compare routine in this phase is the same as that in phase 2 except for label changes. Records are compared until a winning record is found. Control is then passed to the output routine where the winning record is sequence-checked before being moved to the output area. Before returning to the compare routine, a check is made to see if the last record in the output file has been processed.

Note: The compare routine uses the registers labeled SAVEA, SAVEB, SAVEC, and SAVED as link registers for a 7-way merge. For a 4-way merge, the link registers are SAVEA and SAVEB. The link address for return to the mainline compare routine depends on the sequence from which a winning record is chosen. For example, if the winning record is from sequence 3, the return point (stored in register SAVEC) is to COMP73+2, COMP63+2, COMP53+2, or COMP43+2.

COMP76, DC-B2

Comparing of records starts at COMP76 for a 7-way merge and continues through the mainline until a winning record is found. A record from sequence 7 is compared with a record from sequence 6. If the record from 7 wins, it is compared with a record from sequence 5; as long as 7 is the winner, it is compared with records from the other available sequences, in turn. If 7 is the winner at COMP71, the sequence 7 record is moved to the output area by means of a branch to SEQCHK (Chart DJ).

The program then returns to this routine to check for a depleted sequence 7 block, and for end of input sequence 7. If sequence 7 block is not depleted, a branch is made to COMP76 to compare the next record in sequence 7 with a record in sequence 6, and so on, until a winner is found. If sequence 7 block is depleted but it is not the end of the sequence, a branch is made to GET7 to read in another block.

If the winning record is:

- 6, branch to COMP65 (Chart DD).
- 5, branch to COMP54 (Chart DE).
- 4, branch to COMP43 (Chart DF).
- 3, branch to COMP32 (Chart DG).

- 2, branch to COMP21 (Chart DG).
- 1, branch to PUT1 (Chart DG).

Other conditions will alter the flow through the mainline compares. For example, if the end of sequence 6 is reached, the branch code at B75 is set to 15 (unconditional branch). The program then branches to compare 7:5.

If, after moving the sequence 7 winning record to the output area, the end of sequence 7 is reached, control returns to USTOP7 to close all compares for sequence 7.

If a record in sequence 7 is compared against 6, 5, and 4, and 4 is determined to be the winning record, the address of COMP74+2 is stored in register SAVED and the routine continues to COMP43. If sequence 4 record is the winner over 3, 2, and 1, 4 is moved to the output area and control is returned to the mainline location stored in register SAVED. Although the sequence 4 record was found to be the winner, the sequence 7 record had already been found to be the winner over 6 and 5 sequence records. Therefore, control is returned to the compare routine at the point where the previous sequence 7 record is compared to a new sequence 4 record. See Figure 41.

COMP65, DD-B2

A record from sequence 6 is compared with a record from sequence 5. If the record from 6 is found to be the winner, it is compared with a record from sequence 4, and so on until a winner is determined.

If the winner is:

- 6, branch to PUT6.
- 5, branch to COMP54 (Chart DE).
- 4, branch to COMP43 (Chart DF).
- 3, branch to COMP32 (Chart DG).
- 2, branch to COMP21 (Chart DG).
- 1, branch to PUT1 (Chart DG).

After moving the sequence 6 record to the output area, the compares resume at COMP76 if sequence 6 block is not depleted.

If sequence 6 block is depleted but it is not the end of the sequence, a branch is made to GET6 to fill the sequence 6 input area. If the end of sequence 6 is reached,

control returns to USTOP6 to close all compares for sequence 6.

COMP54, DE-B2

A record from sequence 5 is compared with a record from sequence 4. If the record from 5 is found to be the winner, it is compared with a record from sequence 3, and so on until a winning record is determined.

If the winning record is:

- 5, branch to PUT5.
- 4, branch to COMP43 (Chart DF).
- 3, branch to COMP32 (Chart DG).
- 2, branch to COMP21 (Chart DG).
- 1, branch to PUT1 (Chart DG).

After moving the sequence record 5 to the output area, the compares resume at COMP76 if sequence 5 block is not depleted.

If sequence 5 block is depleted but it is not the end of the sequence, a branch is made to GET5 to fill the sequence 5 input area. If the end of sequence 5 is reached, control returns to USTOP5 to close all compares for sequence 5.

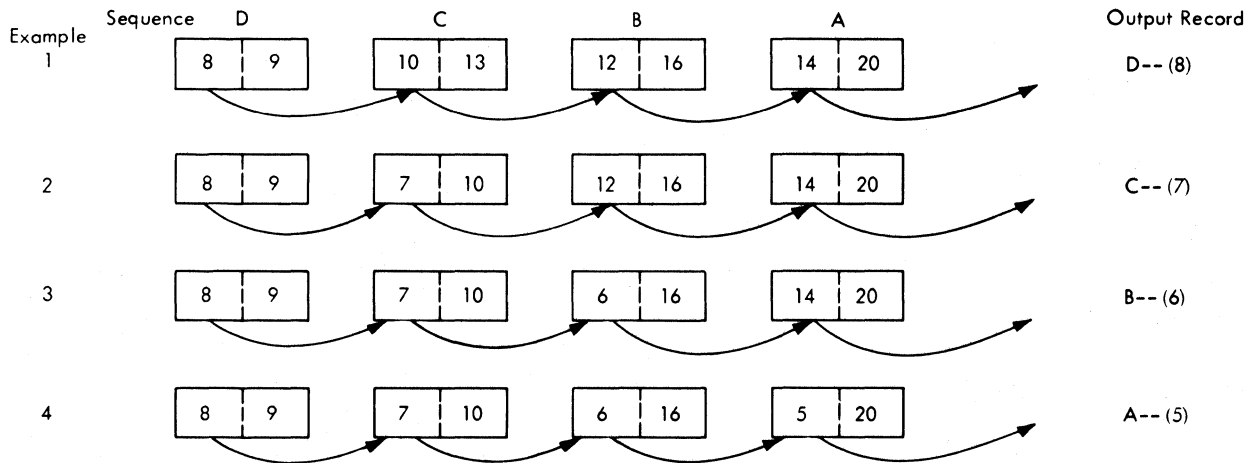
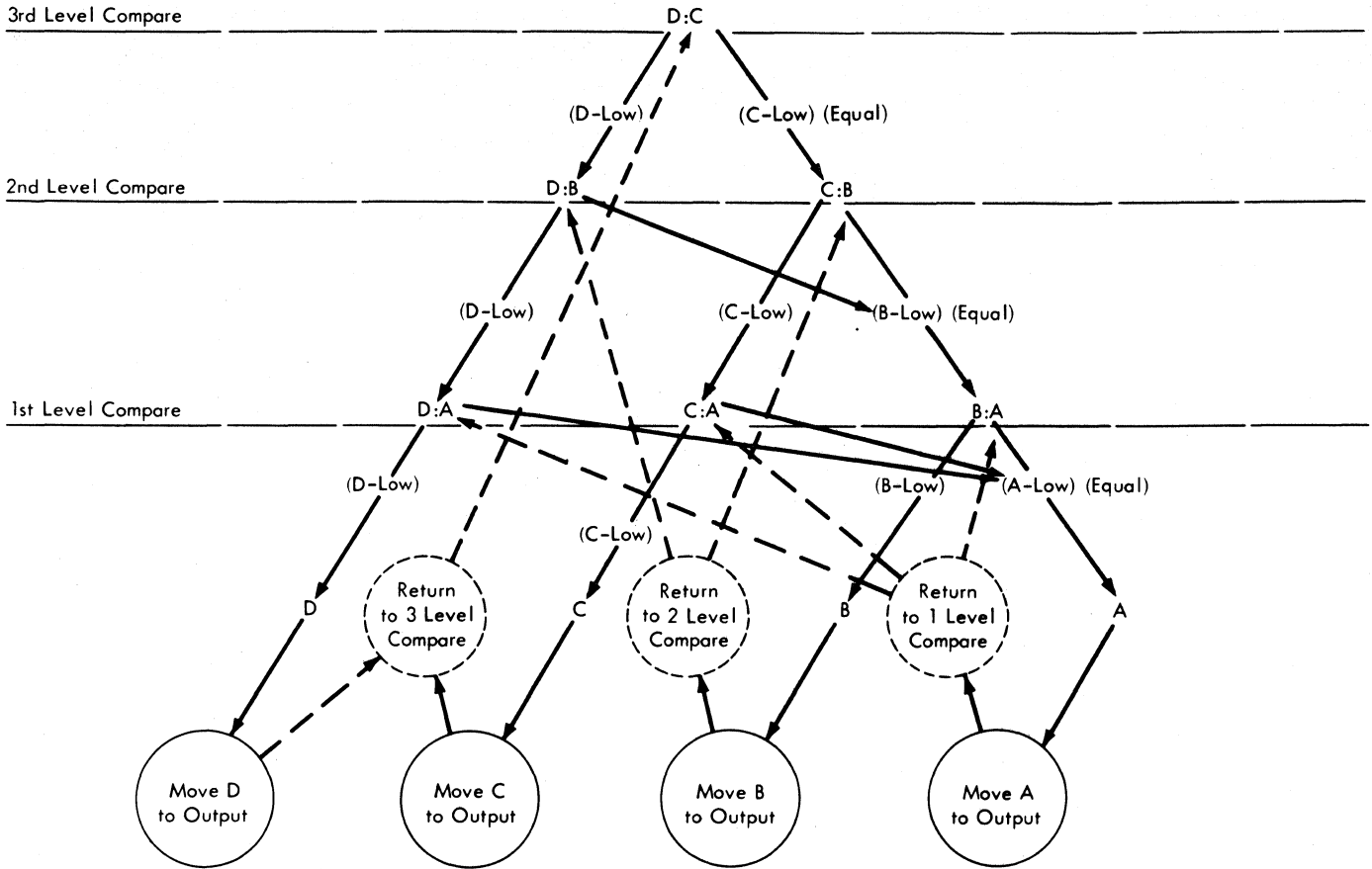
COMP43, DF-B2

Note: For a 4-way merge, comparing of records starts at COMP43 and continues through the mainline in the same manner described under COMP76, until a winning record is found.

A record from sequence 4 is compared with records from sequences 3, 2, and 1. If the winner is:

- 4, branch to PUT4.
- 3, branch to COMP32 (Chart DG).
- 2, branch to COMP21 (Chart DG).
- 1, branch to PUT1 (Chart DG).

After moving the sequence 4 record to the output area, control is returned at the location stored in register SAVED (for 7-way merge) or at COMP43 (for 4-way merge), if sequence 4 block is not depleted.



If D or C is moved to the output area, 3 compares are required before another record can be moved.

If B is moved to the output area, 2 compares are required before another record can be moved.

If A is moved to the output area, 1 compare is required before another record can be moved.

Figure 41. Compare Tree (for a 4-Way Merge)

If sequence 4 block is depleted but it is not the end of the sequence, a branch is made to GET4 to fill the sequence 4 input area. If the end of sequence 4 is reached, control returns to USTOP4 to close all compares for sequence 4.

COMP32, DG-B1

A record from sequence 3 is compared with records from sequences 2 and 1. If the winner is:

- 3, branch to PUT3.
- 2, branch to COMP21.
- 1, branch to PUT1.

After moving the sequence 3 record to the output area, control is returned at the location stored in register SAVEC (for 7-way merge) or to COMP43 (for 4-way merge), if sequence 3 block is not depleted.

If sequence 3 block is depleted but it is not the end of the sequence, a branch is made to GET3 to fill the sequence 3 input area. If the end of sequence 3 is reached, control returns to USTOP3 to close all compares for sequence 3.

COMP21, DG-C3

A record from sequence 2 is compared with a record from sequence 1. If 2 is the winner, branch to PUT2; if 1 is the winner, branch to PUT1.

After moving the sequence 2 record to the output area, control is returned at the location stored in register SAVEB if sequence 2 block is not depleted.

If sequence 2 block is depleted but it is not the end of the sequence, a branch is made to GET2 to fill the sequence 2 input area. If the end of sequence 2 is reached, control returns to USTOP2 to close all compares for sequence 2.

PUT1, DG-E5

When the winning record is from sequence 1 and it has been moved to the output area, control is returned at the location stored in register SAVEA, if sequence 1 block is not depleted. If sequence 1 block is depleted but it is not the end of the sequence, a branch is made to GET1 to fill the sequence 1 input area. If end of sequence 1 is detected, control returns to USTOP1 to close all compares for sequence 1.

COMPUTE INPUT INTERLEAVED DISK ADDRESS,
FIXED-LENGTH RECORDS - DH

The current input disk interleave address is updated by the input interleave factors. The interleave factor RMDR (remainder of: OM divided by BPT) is added to the record number (the R portion of CHHR). If the new record number exceeds the maximum BPT, the next record number is computed by adding the 256 complement of BPT. Refer to Figure 42.

Order of Merge -- 4
 Blocks Per Track -- 5

RMDR -- 0004
 QUOT -- 0000
 BCOMP -- 00FB

	C	H	H	R	
Current Disk Address	01	00	08	01	Block 1
Add (RMDR) Factor		+	00	04	
Record Number Valid (Equal to or Less Than BPT)	01	00	08	05	
Add (Quot) Factor		+	00	00	
Track Number Valid (Equal to or Less Than 09)	01	00	08	05	Block 2
↓					
Current Disk Address	01	00	08	05	Block 2
Add (RMDR) Factor		+	00	04	
Record Number Invalid	01	00	08	09	
Add Complement of BPT (05=00FB)		+	00	FB	
Record Number Valid	01	00	09	04	
Add (Quot) Factor		+	00	00	
Track Number Valid	01	00	09	04	Block 3
↓					
Current Disk Address	01	00	09	04	Block 3
Add (RMDR) Factor		+	00	04	
Record Number Invalid	01	00	09	08	
Add Complement of BPT		+	00	FB	
Record Number Valid	01	00	0A	03	
Add (Quot) Factor		+	00	00	
Track Number Invalid	01	00	0A	03	
Add Complement of 10	00	FF	F6	00	
	02	00	00	03	Block 4

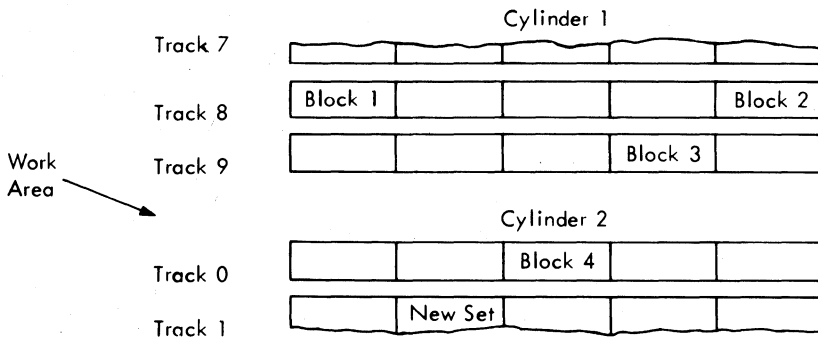


Figure 42. Calculate Interleaved Disk Address

The head number is updated by adding QUOT (quotient of: OM divided by BPT). The new head number is then checked for validity; if it is greater than 9, the next cylinder-head number is calculated by adding the 256 complement of 10.

If the upper limit of the current work area section has been exceeded, the next work area section is accessed. A new interleave disk address is then calculated, based on the lower limit of the new work area section.

LRMDR, DH-B3

The input disk address routine is initialized with the interleave factors for calculating the next disk address of a given sequence (1, 2, 3, 4, 5, 6, or 7 for a 7-way merge; 1, 2, 3, or 4 for a 4-way merge). A branch is then made to CPBPTI.

Note: LRMDR is entered after a block from one of the input sequences has been read into main storage and the interleave factors need not be changed. In these cases, the number of blocks processed (merged to the output file) from the given sequence is equal to or less than the number of blocks in the last sequence passed to phase 3 from phase 2.

LRMDR1, DH-B2

The input disk address routine is initialized with reduced interleave factors for calculating the next disk address of a given sequence (1, 2, 3, 4, 5, or 6 for a 7-way merge; 1, 2, or 3 for a 4-way merge). A branch is then made to CPBPTI.

Note: LRMDR1 is similar to LRMDR except that it is entered after a block has been read into main storage from a sequence other than the last (sequence 7 for a 7-way merge, or sequence 4 for a 4-way merge) and the number of blocks merged to the output file from that sequence is at least one greater than the number of blocks in the last sequence passed to phase 3.

CPBPTI, DH-C2

The next input disk interleave address is calculated with the factors from LRMDR or LRMDR1, as the case may be. The program then returns to the input routine (Chart DB) to store the newly calculated interleave address in the input area table.

OUTPUT ROUTINE, FIXED-LENGTH RECORDS - DJ,DK

When a winning record is determined by the mainline compare loop, this routine is entered for:

- Sequence-checking the output file.
- Data conversion, if specified.
- User exit 32, if specified.
- Updating the main storage output area.
- Moving a record to the main storage output area.

Note: This can be either a winning record or a user-inserted record (via exit 32).

- Writing on tape or disk whenever the main storage output area becomes full.
- Updating the phase 3 record count.
- Open, close, EOV (end-of-volume), and sequence-error conditions.
- Executing end-of-job routine after the last record has been moved to the output area.

Sequence checking is performed by comparing the current winning record (which is about to be moved to the main-storage output area) with the last winning record that was moved to the output area. Because this is the first function of the output routine, the first winning record is not sequence checked. Each record (after the first record) is sequence checked before conversion (if specified) is performed and before the user has access to the record in the output area via exit 32.

Note: The data-conversion routine converts the previous record moved to the output area.

For disk output, the output address is calculated before writing a block of records on the disk output unit. For the first output block, a dummy address 0010

(CHHR) is used to force the upper limit (also a dummy address) to be exceeded. This causes the initial output file extents and the logical unit address to be retrieved, so that a write command can be given to the output file.

For tape output, the initialization routine has relocated routines OP1EOV and EOJTAP to LWRITE+4 and PH3EOJ+4, respectively. Prior to writing the first output block on tape, the initial open condition is executed.

End-of-job routine functions are:

- Initialize to write last output block for disk or tape.
- Initialize to write EOF record and to close output file (disk only).
- Close output file.
- Print end-of-job messages.
- Issue EOJ macro and return control to job control.

The resident checkpoint routine is used to:

- Checkpoint mainline program when open, end-of-volume, close, or sequence-error conditions are found.
- Read in and link to label-linkage routine (LLR).
- Checkpoint the LLR.
- Restore mainline program to main storage and return control to appropriate point in phase 3.

SEQCHK, DJ-B3

For the first winning record, the sequence check, conversion, and exit 32 routines are bypassed; a branch is made to FULOUT. For all subsequent records, the routine continues to CHKSEQ.

CHKSEQ, DJ-B2

The control data field(s) of the winning record is (are) compared with the previous record moved to the output area. If the record to be moved is out of sequence, a branch is made to SEQERR; if no error, the branch is to CONVRT.

SEQERR, DJ-C2

The sequence-error indicator (S-character) is inserted in location OPEN and a branch is made to OPENF (Chart DK) to checkpoint the mainline and read the LLR (label linkage routine) into main storage.

CONVRT, DJ-C1

If data conversion was not specified, the routine continues to EXIT32. If conversion was specified by the user, the record previously moved to the output area is to be reconverted to the format specified in the control statements. A branch-and-link is made to the selected reconversion routine; control is returned at EXIT32.

EXIT32, DJ-E1

If exit 32 was not specified, the routine continues at NXTOR. If exit 32 has been specified, user base register 15 is loaded with the address of the user routine (USADDR) and register 14 (LINK) is loaded with the return address. Control is then passed to the user program. After execution of the user routine, control is returned to the sort program via register 14. For a detailed description of user functions in exit 32, refer to IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.

NXTOR, DJ-G1

The output area address in main storage is updated to move the next winning record, and the switch that permits entry to SEQCHK (for sequence check) is turned on (no-op). The routine then continues to FULOUT.

FULOUT, DJ-C3

A test is made to determine if the main storage output area is full; if it is, a branch is made to LWRITE to write records from the output area onto disk or tape.

If the output area is not full and there is room for at least one more record, the move routine (STR2) is entered to move a record into the output area. This record

may be either a winning record or a user-inserted record. The routine then continues at ZYXWZY-INSERT.

ZYXWZY-INSERT, DJ-E3

If exit 32 was specified and if the record just moved was a user-inserted record, a branch is made to EXIT32 to permit multiple insertions by the user. Without exit 32, this branch is not activated.

A count of the records processed by phase 1 was placed in register 10 (for the 4-way merge program) or in location COUNTR (for the 7-way merge program) during the initialization of phase 3. Every time a winning record is moved to the main storage output area, this count is reduced by one (for the 7-way merge program, the count is loaded in register 10, decremented, and returned to COUNTR). If the result is positive (more records remain), a branch is made back to the mainline compares at NXT1R, NXT2R, NXT3R, etc., as the case may be (Charts DC through DG), to resume processing. If the result is zero (last record has been moved into the output area), the routine returns to SEQCHK to allow the user to perform end-of-job operations via exit 32 (if specified).

Entry to SEQCHK produces a negative value (hexadecimal FFFFFFFF) in register 10. This negative value is provided so that the user program can determine a last-record condition. Control is returned to initiate the end-of-job function.

LWRITE, DJ-C4

After initializing to write a block of records from the output area, the routine continues either to CPBPTO for disk output or to TAPEO for tape output.

For tape output, the sub-routine at label OP1EOV has been relocated to LWRITE+4 during initialization routine. A branch to TAPEO is usually executed to write a block of records on tape. The only exception occurs when the first block is about to be written. Here, as in the case of disk output, the open condition must be performed: The output volume must be rewound (if specified), labels checked and created, etc. For the first output block, a branch is made to OPENF (start of resident checkpoint routine, Chart DK) to read in the label-linkage routine and perform these functions.

CPBPTO, DJ-D5

The next output disk address (CHHR) is calculated; this calculation is similar to that for an input disk address except that the interleave factors are not used. The record number (R) is always incremented by one, and head and record numbers are checked for validity. The newly-computed 2311 address is checked to determine if the upper limit of the output extent has been exceeded. If it has, a branch is made to OPENF (Chart DK) to initiate retrieval of the next set of extents and the logical unit address.

At the start of phase 3, the lower limit is initialized with the address 0010 (CHHR). The upper limit is 0000. For the initial output block, the first calculated disk address is 0011, thus forcing the upper limit to be exceeded. This causes retrieval of the initial output file extents and the logical unit address before a write command is given to the output file.

The disk search address is stored in CHHR, and the actual address is stored in the count field immediately preceding the main storage output area. The R value of the search address (CCHHR) is always one less than the R value of the actual 2311 address. A branch is then made to TAPEO.

TAPEO, DJ-G5

An EXCP macro is issued for IOCS to write a block of records to the output file (tape or disk).

This function is also used to write an end-of-file record on disk (data count of zero) after the last output block has been written in the disk file.

For tape output, when an end-of-reel condition has been detected, a branch is made to CLOSE (Chart DK) to initiate the execution of end-of-volume and open functions for the next output tape volume. When writing the last output tape block of the file, an end-of-volume condition is forced; in this case, the branch is made to CLOSE (Chart DK) to initiate the closing of the output file.

If disk output, or if not end of reel for tape output, the routine continues to PH3EJ1.

PH3EJ1, DJ-J5

If more records remain to be processed, a branch is made to STR2 to move the last winning record to the main-storage output area. For disk output only, when no more records remain to be processed, control is returned to PH3EOJ.

PH3EOJ, DK-B1

The end-of-job functions are initialized for disk or tape:

- For disk output, the number of bytes remaining in the main-storage output area are calculated to determine if the last output block has been written. A byte count of zero indicates that the final output block was already written; the switch at BCLOSE is set so that an unconditional branch is made to OPENF-4 to initiate the final close action.

In either case, the computed data length of the last block (or of the EOF record) is stored in the output count field. The data length is then incremented by eight and stored in the write CCW (WTCCW). The branch at PH3EJ1 is made a no-op to allow re-entry to the end-of-job function at PH3EOJ.

A branch is then made to LWRITE to write either the last output block or the EOF record. If the last block is still to be written, this end-of-job function is re-entered to write the EOF record. If the EOF record is to be written, the switch at BCLOSE initiates the final closing routine.

- For tape output, this function is entered once. (The sub-routine at EOJTAP will have been relocated to PH3EOJ+4 by the initialization routine.) If the ADDROUT "A" option is specified, ten blanks are moved into the main-storage output area (significant only when last block has not been written). This avoids the possibility of a "noise" record because the last output block then contains at least 20 bytes.

The end-of-volume indicator (V-character) is changed to a close indicator (F-character). If last block has been written, a branch is made to CLOSE (Chart DK). If last block has not been written, the switch at BCLOSE is made unconditional branch and a branch is made back to LWRITE+4 to

write the last block. Then, when IOCS returns control to the mainline, a branch is made to CLOSE.

CLOSE, DK-D3

This routine is relocated, during phase 3 initialization, to location OP1EOV+24. The end-of-volume indicator (V-character) is loaded in register 0 for tape EOVS, and a branch is made to RWLABL to read in the LLR and initiate the end-of-volume functions. If end of file, the EOF close indicator (F-character) is loaded in register 0 so that the branch to RWLABL will initiate the close functions.

OPENF, DK-D2

Register 0 is initialized for the condition to be processed:

- Open - O-character
- Close - F-character
- Sequence error - S-character

The routine then continues to RWLABL.

RWLABL, DK-E2

The R value of the disk address (CHHR) for the checkpoint track is initialized to checkpoint the mainline and read in the label-linkage routine. A branch is then made to LABETY.

LABETY, DK-F2

The command code in the channel program (WRMAIN+24) is modified to:

1. Checkpoint mainline (write)
2. Read in label-linkage routine (read)
3. Checkpoint LLR (write)
4. Read-in mainline (read)

Refer to Chart DK for the exit points from LABETY after each of these listed functions.

LABEL-LINKAGE ROUTINE (LLR), FIXED-LENGTH RECORDS - DL

The start of the LLR (label-linkage routine) is identified by the label LINKRT in the program listing. The LLR is written on the checkpoint track during phase 3 initialization; it is then read into main storage (at the end of the supervisor, location RDCHPG) for open, end-of-volume, close, or sequence-error conditions. (A portion of the mainline is checkpointed before the LLR is read in.) When in main storage, the LLR receives control from the resident checkpoint routine through linkage prepared in register 14.

The label-linkage routine initiates open, end-of-volume, and close operations for:

- Disk output - standard file labels
- Tape output - standard file labels, non-standard labels, or no labels.

X31LNK, DL-B2

The condition indicator is obtained from register 0, stored in X31IND, and the indicated branch is made:

- Indicator is 'S' (sequence error) - branch to SEQERROR.
- Indicator is 'V' or 'F' and output is on tape (EOV or EOF) - branch to X31CLS-12.
- Indicator is 'F' and output is on disk (EOV or EOF) - branch to X31CLS+8. (This branch replaced the one for tape output during initialization.)
- Indicator 'O' (open) - continue to IOCSOPEN.

IOCSOPEN, DL-C2

An OPEN macro is issued to open the output file. (This function is executed by the transient IOCS label processing routine.) If exit 31 is specified, IOCS returns control at USRLAB; in either case, this routine is re-entered at X31TS1.

X31TS1, DL-G2

For disk output, the extents are obtained from the DTFSD table and stored in ORADDR-LIMITO. The logical unit address is obtained from the DTF table and placed in the output CCB at OCCB+6.

For tape output, the rewind code in DTFMT is initialized for end-of-volume time and for close time.

This function opens each disk extent or, for tape output, the initial tape volume.

A branch is then made to LABETY (checkpoint routine, Chart DK) to write the LLR on the checkpoint track and to read the mainline into main storage.

USRLAB, DL-E2

Note: This sub-routine is entered from:

1. Transient IOCS label-processing routines.
2. User program associated with exit 31.

The sub-routine is used only if exit 31 has been specified in the MODS control statement. The DTF table (DTFMT or DTFSD) has been initialized to indicate label-address exit. IOCS enters here to permit linkage to the user program via exit 31. Exit 31 functions are:

- create and write non-standard header and trailer labels for tape output.
- build user header and trailer labels for disk or tape output.

The user returns control here to indicate that:

- all non-standard tape header and trailer labels have been created and written, or
- a user header or trailer label has been built and is to be written (with more to follow), or
- the last user header or trailer label has been built and is to be written.

This sub-routine returns control to IOCS with an LBRET macro. When user or non-standard label processing is complete, IOCS then returns to the next sequential instruction after the macro (OPEN, FEOV, or CLOSE) that initiated the label processing operation.

Note: When exit 31 is entered for label processing, the low-order byte of general register 0 contains: O (open), V (end-of-volume), or F (close) to enable the execution of the user label function.

X31CLS-12, DL-C4 (TAPE ONLY)

For EOJ or CLOSE conditions, the volume block count (for standard labels) is inserted in the DTF table to enable IOCS to incorporate it into the standard trailer label. The volume block count is made zero in preparation for the next volume.

X31CLS, DL-D4 (TAPE ONLY)

Location X31IND is tested to determine which of the two conditions exists. For end of volume, a branch is made to X31EVT; for close, the routine continues to IOCSCLOS to close the output file.

X31EVT, DL-E4 (TAPE ONLY)

An FEOV macro is issued and IOCS creates the required output labels.

After the IOCS end-of-volume function is complete, the FEOV switch in the DTFMT table is turned off and the next output volume is opened.

If an alternate drive has been assigned in the SYS001 ASSGN card, automatic volume switching is also done.

A branch is then made to LABETY to write the LLR on the checkpoint track and to read the mainline into main storage.

IOCSCLOS, DL-C3

IOCS closes the output file (disk or tape) and returns control at PRTEOJ.

PRTEOJ, DL-D3

An EXCP macro is issued and the end-of-job messages are printed:

- 7DC4I RECORDS PROCESSED 0000000

- 7DC5I END OF SORT

When IOCS returns control to this sub-routine, an EOJ macro is issued.

SEQERROR, DL-C5

An EXCP macro is issued and the message "7DC2D SEQ. ERROR" is printed on SYSLOG.

Note: If SYSLOG is an IBM 1052 Printer-Keyboard, the operator's reply to this message is either IGNORE or CANCEL. CANCEL results in program cancellation by IOCS. IGNORE permits processing to continue; a branch is made to LABETY to write the LLR on the checkpoint track and to read the mainline into main storage.

If the operator's reply is incorrect (neither IGNORE nor CANCEL in upper or lower case letters), the message "7DC2A INVALID RESPONSE" is printed. The operator must retype the reply to the sequence-error message.

If SYSLOG is not an IBM 1052 Printer-Keyboard, the program is automatically canceled.

PHASE 3 INITIALIZATION, VARIABLE-LENGTH RECORDS - DM

The checkpoint record and the phase 2 constants pertinent to phase 3 are read into main storage and several routines are initialized:

- Mainline compare routine.
- Exit 32 routines.
- Disk or tape output routine.
- Disk or tape label-linkage routine.
- Relocatable routines.

The interleave factors used for the disk input routine are calculated, as are the constants for phase 3 input/output areas.

For disk output, the output routine is executed as shown in the listing. For tape output, the routines at labels OP1EOV and EOJTAP are relocated to TAPOV2+4 and PH3EOJ, respectively, at initialization time.

A check is made to determine if user programming is to be included. Exits available to the user in this phase are 31 and 32.

After initialization is complete, control is passed to the input-data routine.

INTPH3, DM-B1

Job control, employing the system loader, loads phase 3 into main storage following the supervisor. The transfer address is identified by the label INTPH3. Base register 11 is loaded, and registers 2 and 3 are stored. These two registers contain the logical unit address and the disk address (CHHR) of the checkpoint record, respectively. The checkpoint record, created by the assignment phase and updated by phases 1 and 2, is read into main storage (starting at CKPTRD) by the channel program.

PH3CON, DM-C1

The channel program is modified to read phase 3 constants from the checkpoint track into main storage, starting at IPTCCB and continuing through FRADDR for a 6-way merge or CRADDR for a 3-way merge. A branch-and-link is made to CHEKPT to read these constants into main storage and control is returned at TESTEQ.

TESTEQ, DM-D1

If multiple control data fields are specified in the SORT control card, the equal routine is required and initialization of the mainline compare branches is bypassed. If the equal routine is not required, the branch instructions following each mainline compare are initialized accordingly. For example, if a record from sequence 4 is equal to a record from sequence 3; the next compare is between records from sequences 3 and 2, not sequences 4 and 2.

The sequence error routine is initialized according to the unit assignments. If SYSLOG is a 1052 Printer-KeyBoard the operator may either ignore a sequence error and continue processing or cancel the job when a sequence error is detected. If SYSLOG is not a 1052 Printer-KeyBoard, a sequence error automatically cancels the job.

ITCOMP, DM-E1

The mainline compare loops are initialized for length, location, and collating sequence of control data field 1. This information is contained in the 96-byte table (CF1LCT) in the checkpoint record.

OPTION, DM-G1

A test is made to determine if exit 32 is specified and, if it is, a branch is made to bypass the EXIT32 initialization. If exit 32 is not specified, the output routine is initialized to bypass the exit.

RECORD0, DM-H1

The number of records processed by phase 1 (RECPH1) is obtained from the checkpoint record. If the number of records processed is greater than zero, a branch is made to INTLEAVE; if zero, the branch is to PRTEOJ (Chart DW).

INTLEAVE, DM-B3

The interleave factors are calculated and the input disk address routine is initialized.

The order of merge to be used in phase 3 (number of remaining sequences) is divided by BPT (number of sort blocks per 2311 track). The quotient and remainder are the initial interleave factors for accessing the input to phase 3. The OM is reduced by one and the process is repeated to obtain the reduced interleave factors. If OM is equal to one, reduced factors are not calculated.

The input disk address routine is initialized by storing the interleave factors in LRMDR+1 and LQOUT+1, and the reduced interleave factors in LRMDR1+1 and LQOUT1+1.

TAPLLR, DM-C3

For disk output, this function is bypassed by branching to OUTDSK. For tape output, the CCB (OCCB) and the CCW (WTCCW) are initialized to write tape. The routines for open and EOF (OP1EOV), and the routine

to write the last output tape block and to link to close file (EOJTAP), are relocated and initialized.

The LLR (label-linkage routine) is initialized to set the rewind code for close time in the DTF table immediately after the OPEN has been executed. This rewind code is in effect at end-of-volume time (multi-volume file). For instance, if the user specifies UNLD (unload), each volume is rewound and unloaded at EOJ time, and the next volume is rewound (see Figure 43).

NO ALTERNATE DRIVE Remarks	USER'S REWIND SPECIFICATIONS		ALTERNATE DRIVE/S Remarks
	OPEN	CLOSE	
The volume is rewound and unloaded. The next volume is rewound.	RWD	RWD	The volume is rewound. The next volume is rewound.
The volume is rewound and unloaded. The next volume is rewound.	RWD	UNLD	The volume is rewound and unloaded. The next volume is rewound.
The volume is rewound and unloaded. The next volume is not rewound.	RWD	NORWD	The volume is not rewound, and the next volume is not rewound.
The volume is rewound and unloaded. The next volume is rewound.	NORWD	RWD	The volume is rewound, and the next volume is rewound.
The volume is rewound and unloaded. The next volume is rewound.	NORWD	UNLD	The volume is rewound and unloaded. The next volume is rewound.
The volume is rewound and unloaded. The next volume is not rewound.	NORWD	NORWD	The volume is not rewound, and the next volume is not rewound.

Figure 43. Rewind Action Taken at End-of-Volume Time for Multi-Volume Tape Files

If nonstandard labels or no labels are specified, the tape mark option is initialized. This applies only to an OPEN condition. In this case, the user has the option of inserting or not inserting a tape mark prior to the first record of each output volume. In the case of standard labels, a tape mark is written by IOCS.

If nonstandard or additional user header and trailer labels have been specified, the DTF is initialized to enable IOCS to link to the LLR which, in turn, links to the user through exit 31 to process labels.

At this point, the initialization steps for tape output only are complete and a branch is made to CPLINK to continue initializing for disk and tape.

OUTDSK, DM-D4

Portions of Phase 3 are initialized for disk output:

- Output channel program to verify each output block as it is written on disk, if VERIFY option is specified.
- Disk DTFSD for additional user labels, if exit 31 is specified.
- Label-linkage routine, by overlaying DTFSD (disk) onto DTFMT (tape).

CPLINK, DM-F3

The label-linkage routine is written on the checkpoint track.

Whenever an OPEN, EOJ (tape only), CLOSE, or sequence error condition exists, the label-linkage routine is read into main storage and the condition is processed.

RTNEQ, DM-G3

The relocater routine is initialized to include the equal routine (if number of control data fields is greater than one). If user's format does not require conversion, phase 3 is initialized to bypass the linkage to the conversion routine.

A branch-and-link is then made to the relocater routine (Chart FA) to initialize and relocate the reconversion and equal routines. Control is returned by the relocater at START.

START, DM-J3

The output block length and information pertinent to user-programmed routines in phase 3 are obtained from the checkpoint record, and stored at BKLOUT-PHEX34.

The relocater routine has stored, in the full word constant RLISA, the main storage address of the first available byte to be used by the phase 3 input/output areas. The remaining initialization routines are relocated, starting at the main storage address contained in RLISA. This is necessary because, in the case of disk output, initialization of the output count field (key length and data length) could

possibly destroy unexecuted instructions in routines starting at OUTAPE and ending at PHEX34.

START1, DM-B5

The input areas are allocated for reading the input sequences into main storage. The number of input areas required is equal to the number of sequences to be merged (PH310M) in phase 3. Each input area is equal in length to the sort block size (SORTL) plus an overflow area equal to the maximum record length (LMAX) minus one. The overflow area is located in front of each input area and is used in compacting split input records.

The starting and ending addresses for each input area are calculated and stored in the constants AEND through FBEGIN (CBEGIN for a 3-way merge). For example, AEND contains the end address (address of last byte) of input sequence 1 block (sequence A from phase 2), and ABEGIN contains the starting address of input sequence 1 block. If disk output has been specified, the 8-byte count field (the field immediately following the last input area and adjacent to the first byte of the output area) is initialized. The count field becomes part of each output block and is written out with the data portion and key portion (if specified). The output CCW is initialized with the actual data count for either disk or tape output.

OUTAPE, DM-C5

The output area addresses are calculated and stored; the starting address in OBEGIN, the end address in OUTEND and OUTEND1. The data count for the input CCW is initialized with the sort block length.

The size of the output area depends on the user specification in the SORT control statements.

PH3MRG, DM-D5

Phase 3 is initialized to execute during the final pass:

- a 1, 2, 3, 4, 5, or 6-way merge (for 6-way merge)

- a 1, 2, or 3-way merge (for 3-way merge)

A test is made for any user programming (exit 31 and 32). If a user program phase is to be fetched from the core image library, it is necessary to:

- Extract the origin address of the user program (stored in PHEX34) and insert it in USADDR.
- Load user base register (15) with user program origin address (USADDR).
- Load user link register (14) with the phase 3 return address (USTOP1).
- Execute FETCH macro to load the user program.

When job control has loaded the user program phase into main storage, control is transferred to the user routine so that it may be initialized, if so desired, before phase 3 is executed. If the user program is to be initialized at this time, the information in Figure 44 will be available in the indicated registers at user program fetch time.

The user must be able to return to the sort program, via the address (USTOP1) stored in general register 14, to open the mainline. If user programming is not specified, only the branch to USTOP1 is executed and the remainder of the routine is bypassed.

INPUT ROUTINE, VARIABLE-LENGTH RECORDS - DN

After completing the initialization routine, the input areas are filled with records from the input portion of the disk work area. Phase 3 designates the input sequences 1, 2, 3, 4, 5, and 6, respectively, for a 6-way merge only; in the case of a 3-way merge, input sequences are designated 1, 2, and 3.

At the start of phase 3, mainline compares are opened for sequence 1. A channel program and CCB are prepared to read the first block of sequence 1. After a block is read into an input area, the disk address of the next sequence 1 block is calculated (see compute-disk-address routine, Chart DT). The new disk address is stored in the input address table (ARADDR).

USTOP3, DN-B2

The mainline compares are initialized to open sequence 3, channel programs are prepared with the main storage input area address contained in CBEGIN, and a sequence 3 block is read into the input area.

If the interleave factors need not be reduced, a branch is made to LRMDR (Chart DT) to calculate the next input interleave address; if the interleave factors are to be reduced, the next input address is calculated at LRMDR1 (Chart DT). Upon return to this routine, the calculated address is stored in CRADDR.

For more than a 3-way merge, a branch is made to USTOP4; for a 3-way merge, the branch is to COMP32 (Chart DS).

At the end of sequence 3, this routine is entered at USTOP3 to close the mainline compares for this sequence. A branch is then made to the location stored in register SAVEC, depending on the current sequence and the order of merge. These locations are listed in the tables on Chart DN.

USTOP4, DN-B2

The mainline compares are initialized to open sequence 4, channel programs are prepared with the main storage input area address contained in DBEGIN, and a sequence 4 block is read into the input area.

If the interleave factors need not be reduced, a branch is made to LRMDR (Chart DT) to calculate the next input interleave address; if the interleave factors are to be reduced, the next input address is calculated at LRMDR1 (Chart DT). Upon return to this routine, the calculated address is stored in DRADDR.

For more than a 4-way merge, a branch is made to USTOP5; for a 4-way merge, the branch is to COMP65 (Chart DP).

At the end of sequence 4, this routine is entered at USTOP4 to close the mainline compares for this sequence. A branch is then made to COMP65 (Chart DP).

USTOP5, DN-B2

The mainline compares are initialized to open sequence 5, channel programs are prepared with the main storage input area

address contained in EBEGIN, and a sequence 5 block is read into the input area.

If the interleave factors need not be reduced, a branch is made to LRMDR (Chart DT) to calculate the next input interleave address; if the interleave factors are to be reduced, the next input address is calculated at LRMDR1 (Chart DT). Upon return to this routine, the calculated address is stored in ERADDR.

For more than a 5-way merge, a branch is made to USTOP6; for a 5-way merge, the branch is to COMP65 (Chart DP).

At the end of sequence 5, this routine is entered at USTOP5 to close the mainline compares for this sequence. A branch is then made to COMP65 (Chart DP).

USTOP6, DN-B2

The mainline compares are initialized to open sequence 6, channel programs are prepared with the main storage input area address contained in FBEGIN, and a sequence 6 block is read into the input area.

A branch is made to LRMDR (Chart DT) to calculate the next input interleave address. Upon return to this routine, the calculated address is stored in FRADDR and a branch is made to COMP65 (Chart DP).

At the end of sequence 6, this routine is entered at USTOP6 to close the mainline compares for this sequence. A branch is then made to COMP65 (Chart DP).

MAINLINE COMPARE ROUTINE, VARIABLE-LENGTH RECORDS - DP, DQ, DR, DS

The mainline compare routine in this phase is the same as that in phase 2 except for label changes. Records are compared until a winning record is found. Control is then passed to the output routine where the winning record is sequence-checked before being moved to the output area. Before returning to the compare routine, a check is made to see if the last record in the output file has been processed.

Note: The compare routine uses the registers labeled SAVEA, SAVEB, and SAVEC as link registers for a 6-way merge. For a 3-way merge, the link register is SAVEA. The link address for return to the mainline compare routine depends on the sequence from which a winning record is chosen.

For example, if the winning record is from sequence 3, the return point (stored in register SAVEC) is to COMP63+2, COMP53+2, or COMP43+2.

COMP65, DP-B2

Comparing of records starts at COMP65 for a 6-way merge and continues through the mainline until a winning record is found. A record from sequence 6 is compared with a record from sequence 5. If the record from sequence 6 wins, it is compared with a record from sequence 4; as long as 6 is the winner, it is compared with records from the other available sequences, in turn. If 6 is the winner at COMP61, the sequence 6 record is moved to the output area by means of a branch to SEQCHK (Chart DU).

The program then returns to this routine to check for a split record, for a depleted sequence 6 block, and for end of input sequence 6. If a split record, a branch is made to SPLIT6. If sequence 6 block is not depleted, a branch is made to COMP65 to compare the next record in sequence 6 with a record in sequence 5, and so on, until a winner is found. If sequence 6 block is depleted but it is not the end of the sequence, a branch is made to GET6 to read in another block.

If the winning record is:

- 5, branch to COMP54 (Chart DQ).
- 4, branch to COMP43 (Chart DR).
- 3, branch to COMP32 (Chart DS).
- 2, branch to COMP21 (Chart DS).
- 1, branch to PUT1 (Chart DS).

Other conditions will alter the flow through the mainline compares. For example, if the end of sequence 5 is

reached, the branch code at B64 is set to 15 (unconditional branch). The program then branches to compare 6:4.

If, after moving the sequence 6 winning record to the output area, the end of sequence 6 is reached, control returns to USTOP6 to close all compares for sequence 6.

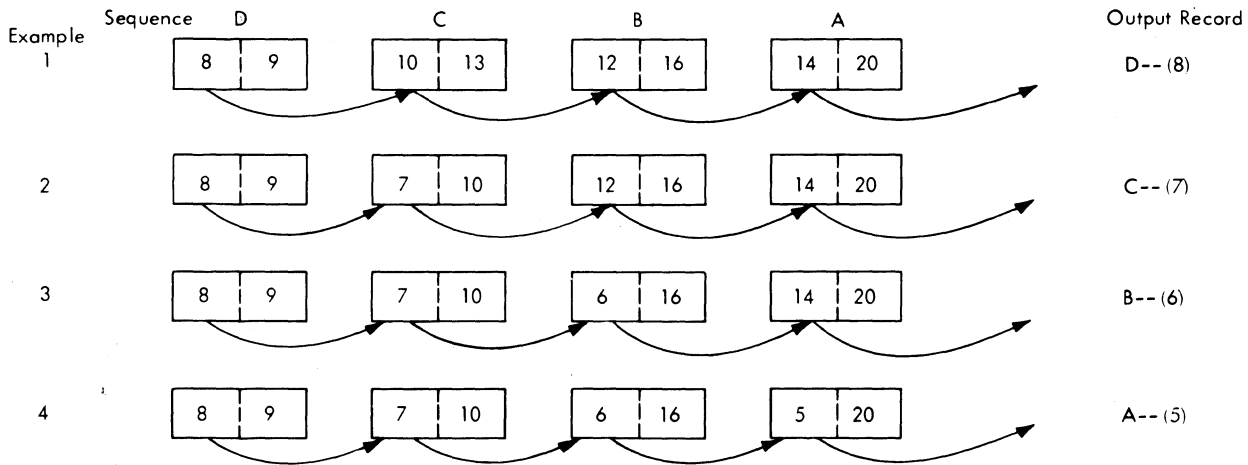
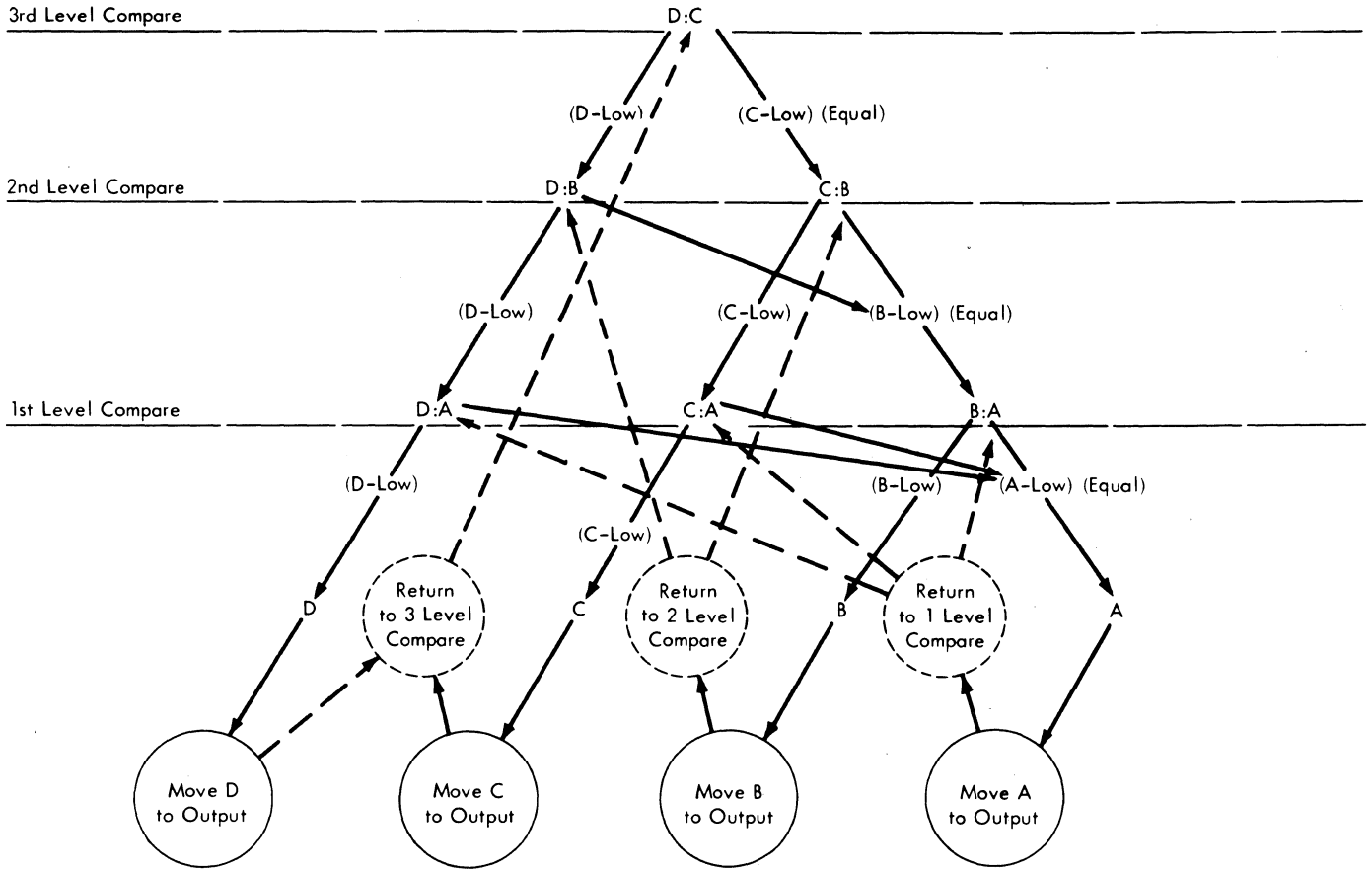
If a record in sequence 6 is compared against 5, 4, and 3, and 3 is determined to be the winning record, the address of COMP63+2 is stored in register SAVEC and the routine continues to COMP32. If sequence 3 record is the winner over 2 and 1, 3 is moved to the output area and control is returned to the mainline location stored in register SAVEC. Although the sequence 3 record was found to be the winner, the sequence 6 record had already been found to be the winner over 5 and 4 sequence records. Therefore, control is returned to the compare routine at the point where the previous sequence 6 record is compared to a new sequence 3 record. See Figure 45.

COMP54, DQ-B2

A record from sequence 5 is compared with a record from sequence 4. If the record from 5 is found to be the winner, it is compared with a record from sequence 3, and so on until a winning record is determined.

If the winning record is:

- 5, branch to PUT5.
- 4, branch to COMP43 (Chart DR).
- 3, branch to COMP32 (Chart DS).
- 2, branch to COMP21 (Chart DS).
- 1, branch to PUT1 (Chart DS).



If D or C is moved to the output area, 3 compares are required before another record can be moved.

If B is moved to the output area, 2 compares are required before another record can be moved.

If A is moved to the output area, 1 compare is required before another record can be moved.

Figure 45. Compare Tree (for a 4-Way Merge)

After moving the sequence record 5 to the output area, the compares resume at COMP65 if sequence 5 block is not depleted or if next record is not a split record.

If sequence 5 block is depleted but it is not the end of the sequence, a branch is made to GET5 to fill the sequence 5 input area. If the end of sequence 5 is reached, control returns to USTOP5 to close all compares for sequence 5. If a split record condition exists, a branch is made to SPLIT5.

COMP43, DR-B2

A record from sequence 4 is compared with records from sequences 3, 2, and 1. If the winner is:

- 4, branch to PUT4.
- 3, branch to COMP32 (Chart DS)
- 2, branch to COMP21 (Chart DS)
- 1, branch to PUT1 (Chart DS).

After moving the sequence 4 record to the output area, control is returned to COMP54 if sequence 4 block is not depleted or if next record is not a split record.

If sequence 4 block is depleted but it is not the end of the sequence, a branch is made to GET4 to fill the sequence 4 input area. If the end of sequence 4 is reached, control returns to USTOP4 to close all compares for sequence 4. If a split record condition exists, a branch is made to SPLIT4.

COMP32, DS-A1

Note: For a 3-way merge, comparing of records starts at COMP32 and continues through the mainline in the same manner described under COMP65, until a winning record is found.

A record from sequence 3 is compared with records from sequences 2 and 1. If the winner is:

- 3, branch to PUT3.
- 2, branch to COMP21.
- 1, branch to PUT1.

After moving the sequence 3 record to the output area, control is returned at the location stored in register SAVEC (for 6-way merge) or to COMP32 (for 3-way merge), if sequence 3 block is not depleted or if next record is not a split record.

If sequence 3 block is depleted but it is not the end of the sequence, a branch is made to GET3 to fill the sequence 3 input area. If the end of sequence 3 is reached, control returns to USTOP3 to close all compares for sequence 3. If a split record condition exists, a branch is made to SPLIT3.

COMP21, DS-B3

A record from sequence 2 is compared with a record from sequence 1. If 2 is the winner, branch to PUT2. If 1 is the winner, branch to PUT1.

After moving the sequence 2 record to the output area, control is returned at the location stored in register SAVEB if sequence 2 block is not depleted or if next record is not a split record.

If sequence 2 block is depleted but it is not the end of the sequence, a branch is made to GET2 to fill the sequence 2 input area. If the end of sequence 2 is reached, a branch is made to USTOP2 to close all compares for sequence 2. If a split record condition exists, a branch is made to SPLIT2.

PUT1, DS-D5

When the winning record is from sequence 1 and it has been moved to the output area, control is returned at the location stored in register SAVEA, if sequence 1 block is not depleted. If sequence 1 block is depleted but it is not the end of the sequence nor a split record, a branch is made to GET1. If end of sequence 1 is reached, a branch is made to USTOP1 to close all compares for sequence 1. If a split record condition exists, a branch is made to SPLIT1.

SPLIT6, DP-H5

Note: The description of this function also applies to:

- SPLIT5, DQ-H5

- SPLIT4, DR-H5
- SPLIT3, DS-H2
- SPLIT2, DS-H4
- SPLIT1, DS-K5

This routine initializes to move the first part of the split record to the overflow area for the corresponding input area. A branch-and-link is then made to ISPLIT in the output routine. Upon return to this routine, the real output address (SAVPUT) is reloaded in register PUTOUT and a branch is made to GET6 (or GET5, etc., as the case may be) in the input routine. This last branch is made because a split record signals a depleted input area.

COMPUTE INPUT INTERLEAVED DISK ADDRESS,
VARIABLE-LENGTH RECORDS - DT

The current input disk interleave address is updated by the input interleave factors. The interleave factor RMDR (remainder of: OM divided by BPT) is added to the record number (the R portion of CHHR). If the new record number exceeds the maximum BPT, the next record number is computed by adding the 256 complement of BPT. Refer to Figure 46.

The head number is updated by adding QUOT (quotient of: OM divided by BPT). The new head number is then checked for validity; if it is greater than 9, the next cylinder-head number is calculated by adding the 256 complement of 10.

If the upper limit of the current work area section has been exceeded, the next work area section is accessed. A new interleave disk address is then calculated, based on the lower limit of the new work area section.

LRMDR, DT-B3

The input disk address routine is initialized with the interleave factors for calculating the next disk address of a

given sequence (1, 2, 3, 4, 5, or 6 for a 6-way merge; 1, 2, or 3 for a 3-way merge). A branch is then made to CPBPTI.

Note: LRMDR is entered after a block from one of the input sequences has been read into main storage and the interleave factors need not be changed. In these cases, the number of blocks processed (merged to the output file) from the given sequence is equal to or less than the number of blocks in the last sequence passed to phase 3 from phase 2.

LRMDR1, DT-B2

The input disk address routine is initialized with reduced interleave factors for calculating the next disk address of a given sequence (1, 2, 3, 4, or 5 for a 6-way merge; 1 or 2 for a 3-way merge). A branch is then made to CPBPTI.

Note: LRMDR1 is similar to LRMDR except that it is entered after a block has been read into main storage from a sequence other than the last (sequence 6 for a 6-way merge, or sequence 3 for a 3-way merge) and the number of blocks merged to the output file from that sequence is at least one greater than the number of blocks in the last sequence passed to phase 3.

CPBPTI, DT-C2

The next input disk interleave address is calculated with the factors from LRMDR or LRMDR1, as the case may be. The program then returns to the input routine (Chart DN) to store the newly calculated interleave address in the input area table.

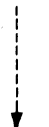
Order of Merge -- 4
 Blocks Per Track -- 5

RMDR -- 0004
 QUOT -- 0000
 BCOMP -- 00FB

Current Disk Address	C	H	H	R	
	01	00	08	01	Block 1
Add (RMDR) Factor		+	00	04	
Record Number Valid (Equal to or Less Than BPT)	01	00	08	05	
Add (Quot) Factor		+	00	00	
Track Number Valid (Equal to or Less Than 09)	01	00	08	05	Block 2



Current Disk Address	C	H	H	R	
	01	00	08	05	Block 2
Add (RMDR) Factor		+	00	04	
Record Number Invalid	01	00	08	09	
Add Complement of BPT (05=00FB)		+	00	FB	
Record Number Valid	01	00	09	04	
Add (Quot) Factor		+	00	00	
Track Number Valid	01	00	09	04	Block 3



Current Disk Address	C	H	H	R	
	01	00	09	04	Block 3
Add (RMDR) Factor		+	00	04	
Record Number Invalid	01	00	09	08	
Add Complement of BPT		+	00	FB	
Record Number Valid	01	00	0A	03	
Add (Quot) Factor		+	00	00	
Track Number Invalid	01	00	0A	03	
Add Complement of 10	00	FF	F6	00	
	02	00	00	03	Block 4

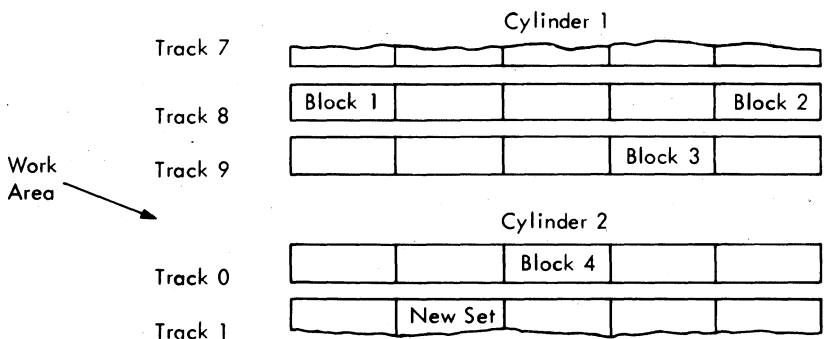


Figure 46. Calculate Interleaved Disk Address

OUTPUT ROUTINE, VARIABLE-LENGTH RECORDS -
DU, DV

When a winning record is determined by the mainline compare loop, this routine is entered for:

- Sequence-checking the output file.
- Data conversion, if specified.
- User exit 32, if specified.
- Updating the main-storage output area.
- Moving a record to the main-storage output area.

Note: This can be either a winning record or a user-inserted record (via exit 32).

- Writing on tape or disk whenever the main-storage output area becomes full.
- Providing for maximum usage of track capacity.
- Updating the phase 3 record count.
- Open, close, EOJ (end-of-volume), and sequence-error conditions.
- Executing end-of-job routine after the last record has been moved to the output area.

Sequence checking is performed by comparing the current winning record (which is about to be moved to the main-storage output area) with the last winning record that was moved to the output area. Because this is the first function of the output routine, the first winning record is not sequence checked. Each record (after the first record) is sequence checked before conversion (if specified) is performed and before the user has access to the record in the output area via exit 32.

Note: The data-conversion routine converts the previous record moved to the output area.

For disk output, the output address is calculated before writing a block of records on the disk output unit. For the first output block, a dummy address 0010 (CHHR) is used to force the upper limit (also a dummy address) to be exceeded. This causes the initial output file extents and the logical unit address to be retrieved, so that a write command can be given to the output file.

For tape output, the initialization routine has relocated routines OP1EOV and

EOJTAP to TAPOV2+4 and PH3EOJ+4, respectively. Prior to writing the first output block on tape, the initial open condition is executed.

End-of-job routine functions are:

- Initialize to write last output block for disk or tape.
- Initialize to write EOF record and to close output file (disk only).
- Close output file.
- Print end-of-job messages.
- Issue EOJ macro and return control to job control.

The resident checkpoint routine is used to:

- Checkpoint mainline program when open, end-of-volume, close, or sequence-error conditions are found.
- Read in and link to label-linkage routine (LLR).
- Checkpoint the LLR.
- Restore mainline program to main storage and return control to appropriate point in phase 3.

SEQCHK, DU-B2

For the first winning record, the sequence check, conversion, and exit 32 routines are bypassed; a branch is made to FULOUT. For all subsequent records, the routine continues to CHKSEQ.

CHKSEQ, DU-C2

The control data field(s) of the winning record is (are) compared with the previous record moved to the output area. If the record to be moved is out of sequence, a branch is made to SEQERR; if no error, the branch is to CONVRT.

SEQERR, DU-D2

The sequence-error indicator (S-character) is inserted in location OPEN and a branch is made to OPENF (Chart DV) to checkpoint the mainline and read the LLR (label linkage routine) into main storage.

CONVRT, DU-D1

If data conversion was not specified, the routine continues to EXIT32. If conversion was specified by the user, the record previously moved to the output area is to be reconverted to the format specified in the control statements. A branch-and-link is made to the selected reconversion routine; control is returned at EXIT32.

EXIT32, DU-F1

If exit 32 was not specified, the routine continues at EOJSW. If exit 32 has been specified, the user base register 15 is loaded with the address of the user routine (USADDR) and register 14 (LINK) is loaded with the return address. Control is then passed to the user program. After execution of the user routine, control is returned to the sort/merge program via register 14. For a detailed description of user functions in exit 32, refer to IBM System/360 Disk Operating System, Sort/Merge Program Specifications, Form C24-3444.

EOJSW, DU-H1

The output area address (PUTOUT) is increased by the length (RLI) of the last record moved to the main-storage output area.

At this point, provision is made for maximum usage of output track capacity. In the initialization routine, the end address of the output area was stored in two locations:

- OUTEND1, which contains the original end address throughout the phase.
- OUTEND, which contains the original end address (OUTEND1) at least until after the first block has been written and

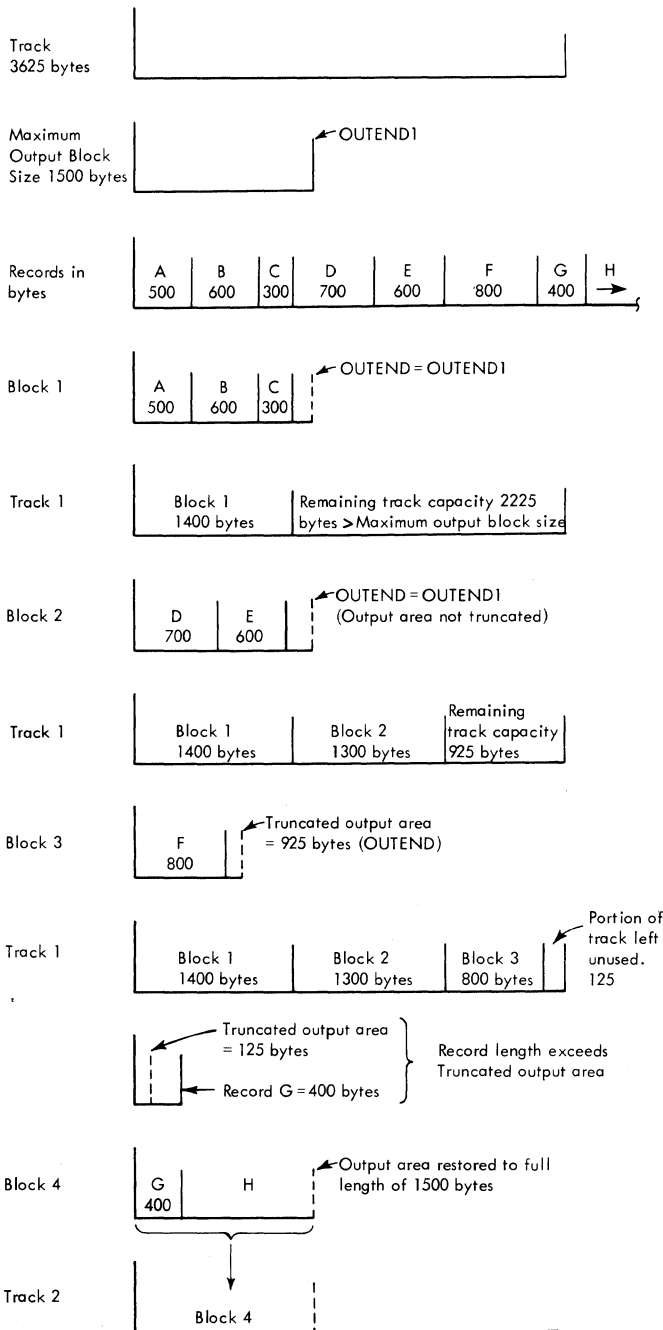
the amount of track capacity remaining has been calculated. If this amount is less than the maximum output block size, the output area is truncated so that it equals the track capacity remaining and the new end address is inserted in OUTEND for testing the next record.

After the first block of a track has been built in the main-storage output area, the track capacity remaining after it is written on disk is calculated. If the track capacity remaining is equal to or greater than the maximum output block size, the block is written and the next record is unconditionally moved to the output area.

If the track capacity remaining is less than the maximum block size, the main-storage output area is truncated (shortened) so that it equals the remaining track capacity, and the new end address of the main-storage output area is placed in OUTEND. The block is written on disk and the next record is unconditionally moved to the main-storage output area.

A test is made to determine if the record unconditionally moved exceeds the shortened output area. If it does not, the record length indicator of the next record to be moved to the main-storage output area is tested to determine if the record will also fit in the output area, and the next output block is built as described in FULOUT. If the unconditionally-moved record does exceed the shortened output area, the output area will be restored to its original length by moving the value of OUTEND1 into OUTEND. The new output block, the first record of which is the unconditionally-moved record, is built and written on the next available track (Figure 47).

The branch at CHKSEQ-3 is made a no-op, after the first record is moved to the main-storage output area, to permit sequence-checking the next record and the routine continues to FULOUT (if the last record has not been processed) or to WLTBLK (if the last record has been processed).



Note: Block sizes in bytes are given for illustration only and do not represent the actual number of bytes consumed in writing the equivalent block.

Figure 47. Maximum Use of Output Track Capacity

FULOUT, DU-B3

The record-length indicator of the record is extracted and stored in the location labeled OUTRLI+2. Register 0, which is used by the move routine, is loaded with

this value. A switch is set (at SPLITR) to inform the move routine that an output record, not a split input record, has been moved. If the output area is filled (cannot contain another record), a branch is made to LWRITE; otherwise, the routine continues to MSPLIT to move the record.

ISPLIT, DU-B4

The output routine is entered at this point from SPLIT6, SPLIT5, SPLIT4, SPLIT3, SPLIT2, or SPLIT1. The split-record indicator (hexadecimal C) of the split record to be moved is erased. Using the index value stored in register 1, the number of bytes of the split record to be moved is calculated. Next, the input overflow area address (the starting byte location to which the first part of the record is to be moved) is computed. A switch is set to inform the move routine that a split record is being moved. A branch is then made to MSPLIT to move the first part of a split input record.

MSPLIT, DU-C3

This function is entered from:

- FULOUT, if the output area is not full.
- ISPLIT, if a split input record is to be moved.
- PH3EJI, if an output block has just been written.

Register 0, which contains the RLI (number of bytes) of the record or split record to be moved, is decremented by one. The variable-length move routine is then initialized and a branch is made to MOVVAR.

MOVVAR, DU-D3

One of two functions is performed at this point. If an output record is to be moved, the record is moved to the main-storage output area. If a split input record is to be moved, the first part of it is moved to the appropriate input overflow area. In either case, register 9 (MREG), which contains the address of the first byte to be moved, is updated to contain the address of the first byte of the next record (if any) within the area from which the record or split record was moved. The routine then continues to SPLITR.

SPLITR, DU-E3

If a split input record was just moved, this switch is an unconditional branch back to GET1, GET2, etc., (via register LINK) to fill the appropriate main-storage input area with a block of records. If an output record was just moved, this switch is a no-op and the routine continues to INSERT.

INSERT, DU-F3

If exit 32 was specified and if the record just moved was a user-inserted record, a branch is made to EXIT32 to permit multiple insertions by the user. Without exit 32, this branch is not activated.

A count of the records processed by phase 1 was placed in register 10 (for the 3-way merge program) or in location COUNTR (for the 6-way merge program) during the initialization of phase 3. Every time a winning record is moved to the main-storage output area, this count is reduced by one (for the 6-way merge program, the count is loaded in register 10, decremented, and returned to COUNTR). If the result is positive (more records remain), a branch is made back to the mainline compares at NXT1R, NXT2R, NXT3R, etc., as the case may be (Charts DP through DS), to resume processing. If the result is zero (last record has been moved into the output area), the routine returns to SEQCHK to allow the user to perform end-of-job operations via exit 32 (if specified).

Entry to SEQCHK produces a negative value (hexadecimal FFFFFFFF) in register 10. This negative value is provided so that the user program can determine a last-record condition. Control is returned to initiate the end-of-job function.

WTLBLK, DU-B5

The entry to this function is at WTLBLK+8 and it is used only when disk output is specified. Entry is made from FULOUT-4 after the last record has been processed in the main-storage output area. A switch is set at PH3EJ1 so that the routine enters the end-of-job functions at PH3EOJ after the last output block has been written. The routine now continues to LWRITE.

LWRITE, DU-C5

Functions of this sub-routine are to:

- Write an output block when the main-storage output area is full.
- Write the last output block.
- Write an end-of-file record for disk output.

The number of bytes to be written in the current output block is computed. This data count is then inserted into the BLI (block-length indicator) of the output block in main storage. For disk output, the data count is also inserted into the DLDL portion of the count field; the data count is incremented by eight and inserted in the write CCW. For tape output, the actual data count is inserted into the write CCW.

For disk output, the 2311 track capacity sub-routine calculates the portion of a 2311 track that will be used in writing the current output block. The method used is to multiply the data count by 537 and divide the product by 512. The result is then incremented by 61 to give the total number of bytes on a track that will be used to write the current block. A test is then made to determine if the records in the output area will fit on the current track; if not, the track capacity is restored to 3633 and the end-of-track indicator (FF) is set. The area remaining on the track is then calculated and stored in TKLEFT.

A calculation and test are now made to determine if the amount of track left is less than the maximum output block size. If so, the output area is truncated to equal the amount of track left (for the reason described in the text under EOJSW). This lower value for the end address of the output area is stored in OUTEND. If this amount of track left is equal to or greater than the maximum output block size, the original end address of the output area is left in OUTEND. Whenever the output area is truncated, it will be restored to its original length at EOJSW (if the record moved does not fit as the first record in the truncated area) after the last block is written on the current track.

The routine then continues to CPBPTO.

For tape output, the sub-routine at label OP1EOV has been relocated to TAPOV2+4 during the initialization routine. A branch to TAPEO is usually executed to write a block of records on tape. The only exception is when the first block is about

to be written. Here, as in the case of disk output, the initial open condition must be performed: The output volume must be rewound (if specified), labels checked and created, etc. For the first output block, a branch is made to OPENF (start of resident-checkpoint routine, Chart DV) to read in the label-linkage routine and to perform these functions.

CPBPTO, DU-D4

The next output disk address (CHHR) is calculated; this calculation is similar to that for an input disk address except that the interleave factors are not used. The record number (R) is always incremented by one, and head and record numbers are checked for validity. The newly-computed 2311 address is checked to determine if the upper limit of the output extent has been exceeded. If it has, a branch is made to OPENF (Chart DV) to initiate retrieval of the next set of extents and the logical unit address.

At the start of phase 3, the lower limit is initialized with the address 0010 (CHHR). The upper limit is 0000. For the initial output block, the first calculated disk address is 0011, thus forcing the upper limit to be exceeded. This causes retrieval of the initial output file extents and the logical unit address before a write command is given to the output file.

The disk search address is stored in CHHR, and the actual address is stored in the count field immediately preceding the main-storage output area. The R value of the search address (CCHHR) is always one less than the R value of the actual 2311 address. A branch is then made to TAPEO.

TAPEO, DU-G5

An EXCP macro is issued for IOCS to write a block of records to the output file (tape or disk).

This function is also used to write an end of file record on disk (data count of zero) after the last output block has been written in the disk file.

For tape output, when an end-of-reel condition has been detected, a branch is made to CLOSE (Chart DV) to initiate the execution of end-of-volume and open functions for the next output tape volume. When writing the last output tape block of

the file, an end-of-volume condition is forced; in this case, the branch is made to CLOSE (Chart DV) to initiate the closing of the output file.

If disk output, or if not end of reel for tape output, the routine continues to PH3EJ1.

In either case, the computed data length of the last block, or of the EOF record, is stored in the output count field. The data length is then incremented by eight and stored in the write CCW (WTCCW). A switch is set at PH3EJ1 (no-op) to allow re-entry to the end-of-job functions at PH3EOJ.

PH3EJ1, DU-J5

If more records remain to be processed, a branch is made to MSPLIT (3-way program) or MSPLIT-8 (6-way program) to move the last winning record to the main-storage output area. For disk output only, when no more records remain to be processed, control is returned to PH3EOJ.

PH3EOJ, DV-B1

Note: For disk output, the last block has been written before this sub-routine is entered.

The switch at BCLOSE is set so that an unconditional branch is made to OPENF-4 to initiate the final close action, and a branch is made to LWRITE. After the EOF record has been written, the open indicator at OPEN is changed to an 'F' (signaling a close operation). A branch is then made to OPENF to close the output file.

For tape output, this function is entered once. (The sub-routine at EOJTAP will have been relocated to the label PH3EOJ+4 by the initialization routine.) The number of bytes contained in the final output block is calculated and inserted in the channel program. The write routine is modified to force an end-of-volume condition and the end-of-volume indicator at EOJ is changed to an 'F'. A branch is then made to LWRITE to write the last output block and close the file.

CLOSE, DV-C4

This routine is relocated, during phase 3 initialization to location OP1EOV+24. The end-of-volume indicator (V-character) is loaded in register 0 for tape EOVS, and a branch is made to RWLABL to read in the LLR and initiate the end-of-volume functions. If end of file, the EOF close indicator (F-character) is loaded in register 0 so that the branch to RWLABL will initiate the close functions.

OPENF, DV-C3

Register 0 is initialized for the condition to be processed:

- Open - O-character
- Close - F-character
- Sequence error - S-character

The routine then continues to RWLABL.

RWLABL, DV-D3

The R value of the disk address (CHHR) for the checkpoint track is initialized to checkpoint the mainline and read in the label-linkage routine. A branch is then made to LABETY.

LABETY, DV-E3

The command code in the channel program (WRMAIN+24) is modified to:

- Checkpoint mainline (write)
- Read in label-linkage routine (read)
- Checkpoint LLR (write)
- Read-in mainline (read)

Refer to Chart DV for the exit points from LABETY after each of these listed functions.

LABEL-LINKAGE ROUTINE (LLR),
VARIABLE-LENGTH RECORDS - DW

The start of the LLR (label-linkage routine) is identified by the label LINKRT in the program listing. The LLR is written on the checkpoint track during phase 3 initialization; it is then read into main storage (at the end of the supervisor, location RDCHPG) for open, end-of-volume, close, or sequence-error conditions. (A portion of the mainline is checkpointed before the LLR is read in.) When in main storage, the LLR receives control from the resident checkpoint routine through linkage prepared in register 14.

The label-linkage routine initiates open, end-of-volume, and close operations for:

- Disk output - standard file labels
- Tape output - standard file labels, non-standard labels, or no labels.

X31LNK, DW-B2

The condition indicator is obtained from register 0, stored in X31IND, and the indicated branch is made:

- Indicator is 'S' (sequence error) - branch to SEQERROR.
- Indicator is 'V' or 'F' and output is on tape (EOV or EOF) - branch to X31CLS-12.
- Indicator is 'F' and output is on disk (EOV or EOF) - branch to X31CLS+8. (This branch replaced the one for tape output during initialization.)
- Indicator 'O' (open) - continue to IOCSOPEN.

IOCSOPEN, DW-C2

An OPEN macro is issued to open the output file. (This function is executed by the transient IOCS label processing routine.) If exit 31 is specified, IOCS returns control at USRLAB; in either case, this routine is re-entered at X31TS1.

X31TS1, DW-G2

For disk output, the extents are obtained from the DTFSD table and stored in ORADDR-LIMITO. The logical unit address is obtained from the DTF table and placed in the output CCB at OCCB+6.

For tape output, the rewind code in DTFMT is initialized for end-of-volume time and for close time.

This function opens each disk extent or, for tape output, the initial tape volume.

A branch is then made to LABETY (checkpoint routine, Chart DV) to write the LLR on the checkpoint track and to read the mainline into main storage.

USRLAB, DW-E2

Note: This sub-routine is entered from:

1. Transient IOCS label-processing routines.
2. User program associated with exit 31.

The sub-routine is used only if exit 31 has been specified in the MODS control statement. The DTF table (DTFMT or DTFSD) has been initialized to indicate label-address exit. IOCS enters here to permit linkage to the user program via exit 31. Exit 31 functions are:

- create and write non-standard header and trailer labels for tape output.
- build user header and trailer labels for disk or tape output.

The user returns control here to indicate that:

- all non-standard tape header and trailer labels have been created and written, or
- a user header or trailer label has been built and is to be written (with more to follow), or
- the last user header or trailer label has been built and is to be written.

This sub-routine returns control to IOCS with an LBRET macro. When user or non-standard label processing is complete, IOCS then returns to the next sequential instruction after the macro (OPEN, FEOV, or CLOSE) that initiated the label processing operation.

Note: When exit 31 is entered for label processing, the low-order byte of general register 0 contains O (open), V (end-of-volume), or F (close) to enable the execution of the user label function.

X31CLS-12, DW-C4 (TAPE ONLY)

For EOVS or CLOSE conditions, the volume block count (for standard labels) is inserted in the DTF table to enable IOCS to incorporate it into the standard trailer label. The volume block count is made zero in preparation for the next volume.

X31CLS, DW-D4 (TAPE ONLY)

Location X31IND is tested to determine which of the two conditions exists. For end of volume, a branch is made to X31EVT; for close, the routine continues to IOCSCLOS to close the output file.

X31EVT, DW-E4 (TAPE ONLY)

An FEOV macro is issued and IOCS creates the required output labels.

After the IOCS end-of-volume function is complete, the FEOV switch in the DTFMT table is turned off and the next output volume is opened. If an alternate drive has been assigned in the SYS001 ASSGN card, automatic volume switching is also done.

A branch is then made to LABETY to write the LLR on the checkpoint track and to read the mainline into main storage.

IOCSCLOS, DW-C3

IOCS closes the output file (disk or tape) and returns control at PRTEOJ.

PRTEOJ, DW-D3

An EXCP macro is issued and the end-of-job messages are printed:

- 7DC4I RECORDS PROCESSED 000000
- 7DC5I END OF SORT

When IOCS returns control to this subroutine, an EOJ macro is issued.

SEQERROR, DW-C5

An EXCP macro is issued and the message "7DC2D SEQ. ERROR" is printed on SYSLOG.

Note: If SYSLOG is an IBM 1052 Printer-Keyboard, the operator's reply to this message is either IGNORE or CANCEL. CANCEL results in program cancellation by IOCS.

IGNORE permits processing to continue; a branch is made to LABETY to write the LLR on the checkpoint track and to read the mainline into main storage.

If the operator's reply is incorrect (neither IGNORE nor CANCEL in upper or lower case letters), the message "7DC2A INVALID RESPONSE" is printed. The operator must retype the reply to the sequence-error message.

If SYSLOG is not an IBM 1052 Printer-Keyboard, the program is automatically canceled.

This phase merges up to four pre-sorted files into one sequential file. This operation is controlled by a MERGE statement in a control card. If a MERGE statement is detected while the assignment phase is processing the control cards, the program initializes for a merge-only operation and phase 4 is fetched.

Input for this phase may be on tape or disk or both; the output may be on tape or disk.

Phase 4 consists of two overlays:

- DSORT401 - open/close routine and its initialization, sequence error routine, and end-of-job messages.
- DSORT402 - merge mainline and its initialization (includes input, compare loops, output, error, end-of-job, and optional routines).

DSORT401 is fetched first, the open/close routine is initialized and written on disk (checkpoint track 2), and DSORT402 is fetched. The merge mainline is initialized and a portion of it is written on disk (checkpoint track 1); part of the mainline resides permanently in storage. DSORT401 is then called in to open the output file, written back on disk, and the non-resident portion of DSORT402 is called in to prepare to open the first input file. The two overlays are then called in, executed, and written back on disk alternately as many times as necessary to perform the initial open and the initial read of all the input files (from 1 to 4, depending on the order of merge).

The merging process is similar to that of phase 3; the major differences are:

- Output records are not interleaved
- The restart feature cannot be used
- Data that is not in binary format is both converted and reconverted

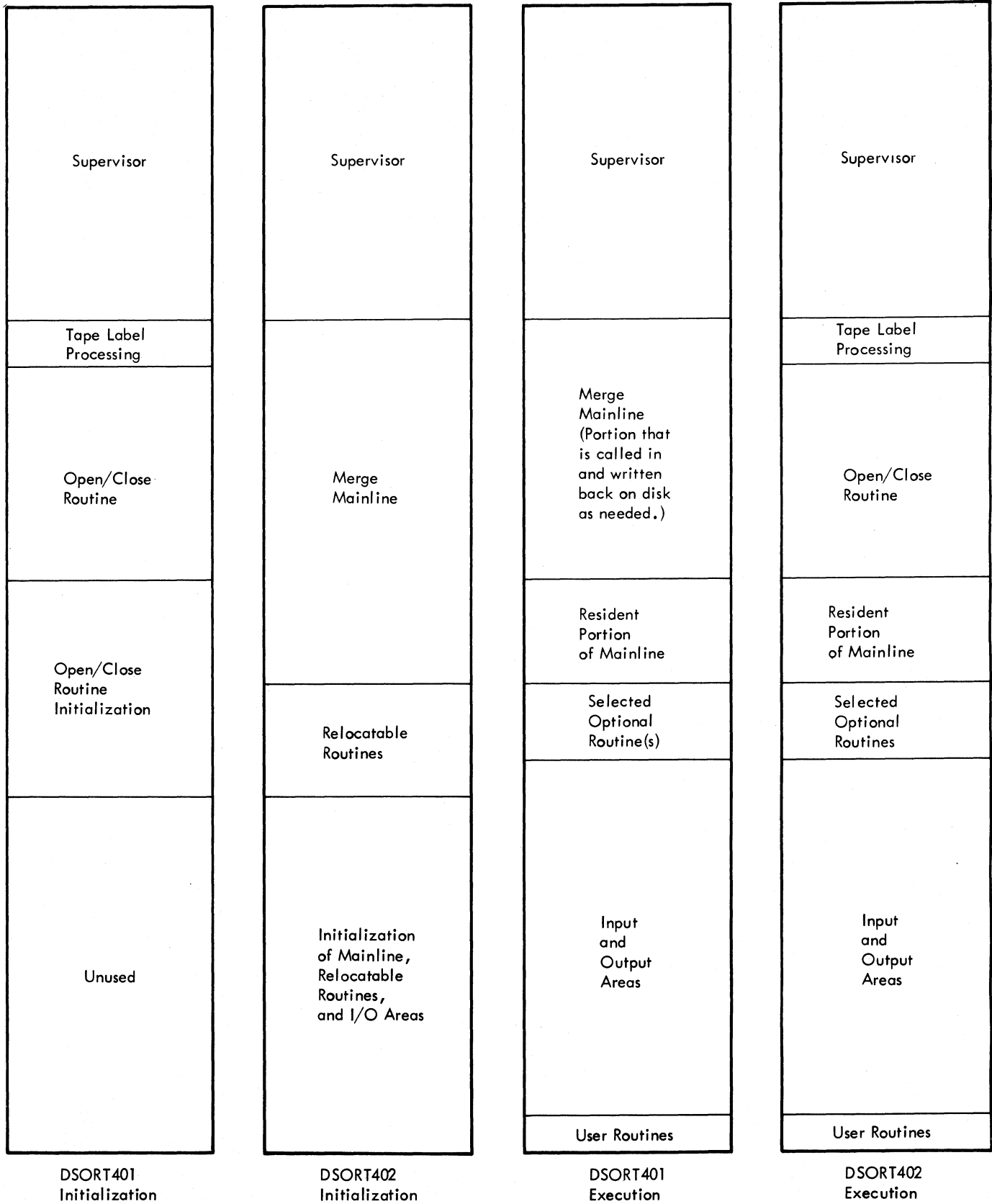
After all the files are open and the input areas filled, records are compared to determine the winner in the user-specified sequence (ascending or descending). The winner is sequence-checked and moved to the output area. As each input area is depleted, it is refilled from its corresponding file until the file is depleted. As the output area is filled, it is written on disk or tape, as specified. End-of-job is reached when all input files have been depleted and the last output block has been written.

Overlay DSORT401 is called in at:

- End of initialization, to open output file and all input files
- End-of-volume, to close and open input volumes
- End-of-file, to close each input file
- Sequence-error, to process the error
- End-of-job, to close the output file and print end-of-job messages

User exits 41 to 45, inclusive, are provided in this phase; 41 and 44 in DSORT401 and 42, 43, and 45 in DSORT402.

Figure 48 is a layout of main storage, showing the two overlays during initialization and execution.



Note: Not drawn to scale.

Figure 48. Phase 4 Main Storage Layout

INITIALIZE OPEN/CLOSE ROUTINE - EA

Using the checkpoint record that was created by the assignment phase, this routine prepares the open/close routine according to number and type of files for use with the merge mainline. Its functions are:

- Read in checkpoint record
- Initialize sequence-error routine for type of message
- Initialize output DTF table for disk or tape and for user and label options
- Initialize input DTF tables (number depends on OM) for disk or tape (or combination of both) and for user and label options
- Write the initialized open/close routine on disk

The routine then fetches the merge mainline (overlay DSORT402).

PANCO, EA-B1

The mainline base register (RG11) is loaded and the contents of registers RG2 and RG3 are stored:

- RG2 - logical unit address of checkpoint - stored in LOGUNIT
- RG3 - address of first track of the work area - stored in TRACK1 and TRACK2

A test is made to determine if both tracks of the work area are in the same cylinder. If they are, the head number is increased by one and the new head address is stored in TRACK2+2; the routine then branches to GETAB. If the tracks are not in the same cylinder, the address of the second track of the work area will be the next cylinder, head zero; this address is stored in TRACK2 and the routine continues to GETAB.

GETAB, EA-E1

The checkpoint record that was created by the assignment phase is read into location FORMAT in main storage for use by phase 4. The address of the user routines and the order of merge are stored in USADDR and OM, respectively.

A test is then made to determine if SYSLOG is a 1052 Printer-Keyboard and, if so, a branch is made to INIT1. If SYSLOG is other than a 1052, the sequence-error routine (Chart EL) is initialized (at instruction RESPONSE) to branch back to the merge mainline through location EXIT in the open/close routine. (Without a 1052, operator response is not possible; message '7DD2D' will be printed and processing will continue.) After modifying instruction RESPONSE, this routine continues to INIT1.

INIT1, EA-H1

Location INOUT is tested to determine if the output is to be on disk and, if not, a branch is made to TAPOUT. For disk output:

- Initialize DTF table FILEOD for disk output.
- Store EOF address in DTF table.
- Change DTF table name from FILEOD to FILEO (this table was assembled as FILEOD to distinguish it from the tape output DTF table, FILEO).
- Test for user exit 44 and, if not specified, turn off the user label address bit in the DTF table.
- Relocate the DTF table to location FILEO.
- Initialize the open/close routine for disk output by modifying the branches at NIEFO, OUTPUT, and OUTDK.

The routine then branches to INIT2.

TAPOUT, EA-J1

The tape output DTF table (FILEO) is initialized for several options, depending on the results of tests:

<u>Option</u>	<u>Test Location</u>	<u>Bit</u>
Rewind first volume	OUTOPT	3
Close unload	OUTOPT	5
Close no-rewind	OUTOPT	7
No tapemark	NOTPMK	2
Non-standard labels	INOUT	2
Unlabeled file	INOUT	1
User exit 44	PH34EX	4

(NOTE: If unlabeled file, user exit 44 not used)

The bit in FILEO + 36 that denotes physical DTF is turned off to convert to logical DTF (to prevent generation of information not needed by this program, the DTF was assembled as DTFPH). A branch is then made to INIT2.

INIT2, EA-A3

Depending on the order of merge, one of the instructions INIT3, INIT4, or INIT5 is made an unconditional branch to CHKPOINT to permit exit from this routine after the required input DTF tables have been

initialized. For example, if the order of merge is two, (1) the instruction at INIT4 is made an unconditional branch, (2) the DTF tables for files A and B (disk or tape, as the case may be) are initialized, and (3) the routine branches to CHKPOINT without initializing the DTF tables for files C and D.

When the branch to CHKPOINT has been initialized, location INPUTA is tested to determine whether file A is on disk or on tape:

- If on disk, load the address of input DTF table FILEAD in register R3, load the address of input DTF table FILEA in R4 (to permit relocating FILEAD later), and branch-and-link to DISK.
- If on tape, load the address of input DTF table FILEA in register R3, load the return address (INIT3) in register LINK, and branch to TAPE.

A similar procedure is followed at INIT3, INIT4, and INIT5 (as required by the order of merge) except that if the file is on tape, INPUTB (and INPUTC and INPUTD, as the case may be) are moved to location INPUTA just before the branch to TAPE. (This permits using just the one location (at INPUTA) to check each tape file, in turn, for the options detailed under TAPE.)

The variations for each file are:

Disk Input Files

<u>Location</u>	<u>Test</u>	<u>DTF Table</u>		<u>BAL to</u>	<u>Return to</u>
		<u>in RG3</u>	<u>in RG4</u>		
INIT3	INPUTB	FILEBD	FILEB	DISK	INIT4
INIT4	INPUTC	FILECD	FILEC	DISK	INIT5
INIT5	INPUTD	FILEDD	FILED	DISK	CHKPOINT

Tape Input Files

<u>Location</u>	<u>Test</u>	<u>DTF Table in RG3</u>	<u>Address in LINK</u>	<u>Branch to</u>	<u>Return to</u>
INIT4	INPUTC	FILEC	INIT5	TAPE	INIT5
INIT5	INPUTD	FILED	CHKPOINT	TAPE	CHKPOINT

DISK, EA-C3

The disk input DTF table FILEAD (or FILEBD, FILECD, or FILEDD) is initialized with EOF address EOFDK, its name is changed to FILEA (or FILEB, FILEC, or FILED, as the case may be), and it is relocated to overlay the tape DTF table for that particular input file. A branch is then made to the address in register LINK (see table following the text at INIT2).

TAPE, EA-C4

The tape input DTF table FILEA (or FILEB, FILEC, or FILED, as the case may be) is initialized for several options depending on the results of tests:

<u>Option</u>	<u>Test Location</u>	<u>Bit</u>
Open rewind	TBKLAB	3
Close unload	TBKLAB	5
Close no-rewind	TBKLAB	7
Non-standard labels	INPUT _n	2
Unlabeled file	INPUT _n	1
User exit 41	PH34EX	7

(Note: If unlabeled file, user exit 41 not used)

The bit in FILE_n+36 that denotes physical DTF is turned off to convert to logical DTF (to prevent generation of information not needed by this program, the DTF was assembled as DTFPH). A branch is then made to the address in register LINK (see table following the text at INIT2).

CHKPOINT, EA-G4

The second track of the work area is formatted and a checkpoint record of the initialized open/close routine is written on disk.

FETCH, EA-H4

The record addresses stored in locations TRACK1 and TRACK2 are set to 1 so that in the merge mainline the checkpoint may be read from, and written back on, disk with

the READ DATA command. Registers RG2 through RG5 are loaded with information for the next overlay (merge mainline):

<u>Register</u>	<u>Load With</u>	<u>Contents</u>
RG2	LOGUNIT	Logical unit address of checkpoint
RG3	TRACK1	Address of first track in disk work area
RG4	TRACK2	Address of second track in disk work area
RG5	LENGTH	Length of the checkpoint record

The merge mainline (overlay DSORT402) is then fetched into main storage for execution.

INITIALIZE MAINLINE - EB

The checkpoint record created by the assignment phase for phase 4 is read into main storage. This routine then:

- Initializes for input files on tape or disk or both.
- Includes or excludes bypass and verify options.
- Provides for conversion and reconversion routines and for user's routines, if required. Exits available to the user are 42, 43, and 45.
- Initializes for fixed- or variable-length records and for variable blocking.
- Initializes for tape or disk output.
- Initializes for ascending or descending sequence and for output sequence checking.
- Initializes and relocates optional routines, if required.
- Computes constants for I/O areas.
- Initializes input and compare loops for order of merge.
- Stores file and record information for the user.
- Fetches user routines (if required).
- Branches to open/close routine for initial open of all files.

After initialization is complete, the program continues to the input routine (Chart EC).

PANCO, EB-B1

The base registers are loaded and the contents of registers RG2 through RG5, which contain information for checkpointing, are stored. (See text for FETCH in the initialization of the open/close routine, Chart EA, for contents of RG2 through RG5). An EXCP macro is then issued and the checkpoint record, which was created by the assignment phase, is read into main storage. The routine then continues to INIT1.

INIT1, EB-D1

Constants are stored for use during the merge:

- Number of volumes for each input file (to be used for tape input only)
- Input block size (maximum for variable-length records)
- Output record length (fixed-length records only)
- Input record length (fixed-length records only)

The last three values listed are also stored in USINF1, USINF2, and USINF2+2, respectively, for user's information in case user exits are specified.

A series of tests are then made to determine which input files, if any, are on tape. If none are, the routine branches through TESTC, TESTB, and TESTA and resumes processing at INIT2. Although the order of merge may be less than 4, all four files are tested; any initialized portions that are not needed will not be used at execution time.

For each file that is on tape, the corresponding CCB is changed to include the read-tape CCW instead of the read-disk channel program. The rest of the initialization for tape input includes:

- The branch to calculate disk addresses in the input routine is made a no-op.
- The instruction in the input routine that moves the data address into the CCW is changed to include the tape CCW instead of the disk CCW.

- The tape indicator (T) is moved to VOL_n
- For standard labels, a hexadecimal 'F0' is placed in LABTYP_n. (This will result in a branch to the END macro instead of to the FEOV macro in the open/close routine at EOv time.)
- For non-standard labels with user exit 41, the number of volumes in the file (VOL_{n+1}) is set to zero and LABTYP_n is initialized as for standard labels.
- For non-standard labels without user exit and for unlabeled files, no modifications are made to LABTYP_n nor to VOL_{n+1}.

The routine then continues to INIT2.

INIT2, EB-E1

A test is made of location KEY to determine if there is a key to be read which signifies that the input files must be on disk only (not mixed) and that only unblocked fixed-length records are being processed. If there is a key, a branch is made to KEYOK; if not, the read CCW is changed to read only data and the error routine is initialized to bypass the function at WLR where the key length is added to the data length. A branch is then made to TSTFORM.

KEYOK, EB-F2

The entry point to the output routine from the end-of-job routine is changed from WRITEV to LWRITE (to write the last block). The test-error routine is initialized to immediately print the wrong-length-record message in case a short block is detected and to return to the input routine immediately. The routine then continues to TSTFORM.

TSTFORM, EB-G1

Tests are made to determine if the data is in a form that must be converted before processing and/or if user exit 42 is specified:

1. Test location FORMAT for conversion. If yes, go to item 3. If no, modify the input routine (Chart EC) so that the branch-and-link to the conversion routine is bypassed. Go to item 2.

2. Test location PH34EX for user exit 42. If yes, go to TESTON. If no, modify the input routine (Chart EC) to bypass the branch-and-link to the user's routine as well as the one to the conversion routine. Then go to TESTON.
3. Test location PH34EX for user exit 42. If yes, go to TESTON. If no, modify the input routine (Chart EC) to bypass only the branch-and-link to the user's routine. Then go to TESTON.

The reason for testing twice for user exit 42 is to initialize the input routine with the proper single instruction that will satisfy any one of three possibilities:

- Bypass the conversion routine.
- Bypass user exit 42.
- Bypass the conversion routine and user exit 42.

TESTON, EB-H1

Tests are made to determine if the data is to be reconverted in the output area to its original format and/or if user exit 43 is specified:

1. Test location FORMAT for reconversion. If yes, go to item 3. If no, modify the output routine (Chart EH) to branch to EXIT to bypass the reconversion routine. Go to item 2.
2. Test location PH34EX for user exit 43. If yes, go to TESTEX. If no, modify the output routine (Chart EH) to branch to UPDTC, thus bypassing both the branch to the reconversion routine and the branch-and-link to the user's routine. Also insert a branch from INSERT to ZYXWZY in the output routine. Finally, insert an unconditional branch from MODF1 to ADD1 in the end-of-job routine (Chart EJ) and go to TESTEX.
3. Test location PH34EX for user exit 43. If yes, go to TESTEX. If no, replace the instruction in the output routine (Chart EH) to bypass the branch-and-link to the user's routine and insert a branch from INSERT to ZYXWZY. Finally, insert an unconditional branch from MODF1 to ADD1 in the end-of-job routine (Chart EJ) and go to TESTEX.

The reason for testing twice for user exit 43 is to initialize the output routine and the end-of-job routine with the fewest possible instructions to satisfy any one of three possibilities:

- Bypass the reconversion routine.
- Bypass user exit 43.
- Bypass the reconversion routine and user exit 43.

TESTEX, EB-J1

If no user exits are required (determined by testing PH34EX), the error routine (Chart ED) is modified to bypass exit 45 and the routine continues to TSBPASS.

If any exits at all are specified, the address of the user's routines is saved and the branch at NOUSER is made a no-op to permit calling in the user routines at the end of the initialization. A test is then made for user exit 45 specifically and, if specified, a branch is made to GOGO; otherwise, the error routine (Chart ED) is modified to bypass this exit and the routine continues to TSBPASS.

TSBPASS, EB-C2

If the BYPASS option has been specified, a branch is made to GOGO; if not, the error routine (Chart ED) is modified to branch past the test for data check. This routine then branches to TSTREC.

GOGO, EB-D3

A bit is turned on in each input file CCB so that control will be returned to this program if a data check is encountered. The routine then continues to TSTREC.

Note: At this point, the processing that began at TESTEX has initialized the mainline so that in case of an unreadable record:

- Without exit 45 and without bypass, phase 4 will not receive control back from IOCS if the particular input file is on disk; if on tape, control will be received if the operator response to the IOCS message is 'ignore'. The error routine will then continue to check for wrong-length record.
- With exit 45 and with bypass, phase 4 will receive control from IOCS and branch-and-link to user 45. When the user returns control, phase 4 will either bypass or process the record,

depending on the user return point in the error routine.

- With bypass but without exit 45, phase 4 will test for a data check and will either bypass the record or process it, depending on the result.

Note that when exit 45 is specified, the BYPASS option is assumed.

TSTREC, EB-E3

Location TYPREC is tested for the type of records to be processed:

- For fixed-length, a branch is made to FIXLGN.
- For variable-length, the output routine (Chart EH) and the error routine (Chart ED) are initialized for variable-length records, and a bit is set on in USINF3 to denote VLR to the user. Then, for disk output only, the instructions for calculating the next disk address for output are initialized for variable-length records and a branch is made to FLDSKO+14. For tape output only, the routine branches to TAPOFX+8.

FIXLGN, EB-F3

The output routine (Chart EH) and the input routine (Chart EC) are initialized for fixed-length records, tape or disk output.

A test is made to determine if L1 (the input record length) is less than L3 (the output record length). If L1 is greater than L3, it indicates that the user wants to shorten records as they are moved to the output area. (Exit 43 must be specified for this purpose when variable-length records are being processed.) L1 will then be used for the move length in the output routine but L3 will be used for incrementing the output area address. The output routine is written for moving L1, so no initialization is necessary in this case and a branch is made to PROS.

In the event that L1 is less than L3, the user wants to lengthen records as they are moved to the output area. The output routine is then initialized for fixed-length records to use L3 for the move length as well as for incrementing the output area address. (Exit 43 must be specified; the user's data will overlay any extraneous matter in the output area that was moved along with the record.) When

processing variable-length records and L1 is less than L3, the output routine will move the entire record (the length specified by the RLI). However, the user must then move the record to the user's work area (via exit 43), lengthen it, and move it back as a user-inserted record. The output routine will then increment the output area address according to the new RLI and move it if there is room in the output area. This procedure is followed because it cannot be known in advance if a variable-length record will fit in the available output area after the user has lengthened it.

The initialization then continues at PROS.

PROS, EB-H3

Location TBKLAB is tested for the type of blocks (fixed or variable) and a branch is made to TAPOFX in case of fixed blocks. Otherwise, the error routine (Chart ED) is initialized to admit variable blocks before continuing at TAPOFX.

TAPOFX, EB-J3

Location INOUT is tested for disk or tape output. For disk output, a branch is made to FLDSKO; for tape, the routine continues at TAPOFX+8.

TAPOFX+8, EB-K3

The mainline is initialized for fixed- or variable-length records, tape output:

- Output routine (Chart EH)
- End-of-job routine (Chart EJ)
- Write Channel Command Word (WTCCW)
- Output Command Control Block (OCCB)

The switch at TPOUT is then turned on (unconditional branch) and a branch is made to ITCOMP.

FLDSKO, EB-J2

The calculations in the output routine for determining the disk output address are initialized with the maximum blocks per track. Note that this is done for fixed-length records only.

FLDSKO+14, EB-K2

The initialization of instructions for disk output is completed and a test is made for the VERIFY option, which is valid only for disk output. If the option is required, the Write CCW is modified to chain to the Verify CCW. The routine then continues to ITCOMP.

ITCOMP, EB-A4

Mainline compare loops are initialized with the length and location of the first control field. This information is contained in the first eight bytes of the 96-byte table starting at location CF1LCT. The length-1 and location-1 of the first control field are then stored in USINF3+1 for the user's information and the output sequence check compare is initialized with the data in CF1LCT+1 and CF1LCT+7.

The instructions in USTOP1, 2, and 3 that determine the branch conditions in the compare loops when results are low, depleted, or equal, are initialized for ascending or descending sequence as required by the data in CF1LCT+7.

CLIFOR, EB-C4

A series of tests is made of location FORMAT to determine which, if any, of the data conversion and reconversion routines is required. The corresponding bit is set in RLCOND for the relocater and in USINF3 for the user's information. A branch is then made to the relocater (Chart FA) to relocate the selected routine, if any, and to initialize the equal routine, if required. The relocation then branches back to this routine of INIT3.

INIT3, EB-E4

The relocater will have stored in location RLISA the address of the first available byte of storage to be used by phase 4 for input and output areas. Using this address, the order of merge (OM), and the input block size (INPBKL), the start and end of each input area is calculated (ABEGIN to AEND, BBEGIN to BEND, etc.). If necessary, the starting point of each area is adjusted to begin at a halfword boundary.

The starting address (OBEGIN) of the output area is calculated after an adjustment is made for it to be at a fullword boundary. OBEGIN is then stored in the write and verify CCW's, the output block size (OUTBKL) is obtained, and the routine continues to TPOUT.

TPOUT, EB-A5

For tape output, this location would have been made an unconditional branch to OUTAPE (see text under TAPOFX+8).

For disk output, a count field is prepared by storing the data length (OUTBKL-key length) in CCOUNT+6 and the key length in CCOUNT+5. OUTBLK is then increased by eight bytes for the count field and the routine continues to OUTAPE.

OUTAPE, EB-C5

The starting address of the output area (OBEGIN), which is in register FILEA, is increased by eight bytes to obtain the address at which the first byte of data will appear in the output area. FILEA is then stored at location OUTEND so that the "move" instruction immediately following can effectively move the address OBEGIN+8 into the write tape CCW.

Using the value in OUTBLK, the end address of the output area is calculated and stored in OUTEND and OUTEND1. The data count for output is inserted in the write and verify CCW's and the data count for input is inserted in the read CCW's for disk and tape.

The byte from the assignment phase table that contains the label information for file A is moved into location LABEL. The corresponding byte for each additional file (B, C, and D, as required by OM) is ANDED, in turn, at location LABEL. If all the

files have the same type labels, the corresponding bit in LABEL will thus be set on; for mixed labels, no bits will be set on. LABEL will then contain a bit configuration that denotes unlabeled, non-standard, standard, or mixed labels for input. LABEL is then tested and the label type is inserted USINF3 for the user's information. The order of merge is determined and also stored in USINF3.

The mainline is then initialized, depending on the order of merge, to fill the input areas and open the corresponding compare loops.

The two instructions at FILLDL are now moved to the beginning of the first input area (at location ABEGIN). This is done so that the execution of these instructions (after fetching the user's routines) will not destroy some part of the initialization routine that has not yet been executed.

NOUSER, EB-F5

When there are no user exits specified, this location is a branch to ABEGIN, where FILLDL is now located. If user exits are specified, however, this location would have been a no-op (see text under TESTEX) and the user routines are now fetched into main storage. When the user returns control to this program through register 14 (which is register LINK), the initialization continues with FILLDL.

FILLDL, EB-H5

The output count field and four blanks (for the BLI in case of variable-length records) are moved to the beginning of the output area and a branch is made to CEVCHK to:

1. Write the initialized mainline on disk,
2. Read in the open/close routine, which will open the output file,
3. Write the open/close routine back on disk, and
4. Read the mainline back into main storage. The program then continues to the input routine (Chart EC) at location TAPE1-2, the address of which is in register RG13.

INPUT ROUTINE - EC

The initial entry to this routine is after the mainline initialization and the opening of the output file. The open/close routine branches back (through CEVCHK) to TAPE1-2 to open the first input file and perform the first read operation. All the files are then opened in turn.

Subsequent entries are made to TAPE1-4 and FIXIP1 from the error routine, and to GET1 from the compare loops.

Note: The labels in this routine are written on Chart EC with the suffix 1 or A and referred to in the text with the same suffixes most of the time. The same logic applies to the similar labels in the listing with suffixes 2, 3, and 4 or B, C, and D, depending on the file in process.

The routine reads in a block of records at a time and branches to the error routine to test for errors. If no errors exist, a branch is made back to this routine at FIXIP1 to provide user exit 42 and/or data conversion, as required. The routine then branches to the compare loops to start merging records.

TAPE1-2, EC-C2

The address of TAPE1 is loaded in register RG13 and the routine continues to TAPE1.

TAPE1, EC-D2

The address of the CCB for the current file (ACCB, BCCB, etc.) is loaded in register MREG, the file indicator is placed in XFILE, and register RG10 is loaded with the address of the input area for that file.

The next instruction in the listing is a no-op except for two occasions:

1. During initial open time, a branch is made to EOFDK in the error routine (Chart ED) to call in the open/close routine via CEVCHK. This branch is then made a no-op until a valid short block is detected at the end of a volume or file.
2. When a valid short block is to be processed, this instruction would have again been made a branch to EOFDK, after which it will be made a no-op.

At all other times, the routine continues to CPIADD for disk input or to TAPEG1 for tape input.

CPIADD, EC-C4

The record number in register RG0 (the current disk address) is increased by one. Location CHHR is then tested for the end-of-track indicator (FF) and, if present, the address is increased to the next track and a test is made to determine if the new head number exceeds nine. If so, the next cylinder number is calculated by adding the 256-complement of ten (HCOMP) to the address in register RG0.

A test is then made to determine if the upper limit of the extent has been exceeded by the new disk address just calculated in register RG0. If not, a branch is made to OKLMTS; if so, location BCERRW+1 is tested to determine if there is a short block to process. If an 'F0' is detected, a branch is made to FROMDK in the error routine (Chart ED) to process the short block; if not, a 'V' for end-of-volume is placed in register 12 and a branch is made to CEVCHK to call in the open-close routine to get the next extent. Control is returned to this routine at GET1 (or GET2, etc., as the case may be).

OKLMTS, EC-F4

The new address in the current extent is stored in ARADDR (or BRADDR, etc.). The new record number is then reduced by one (to obtain the seek address) and stored back in CHHR. A branch is then made to TAPEG1 (or TAPEG2, etc.).

TAPEG1, EC-H2

An EXCP macro is issued to read in a block of records from the file in process. The parameters for the read-in operation are supplied by ACCB, BCCB, CCCB, or DCCB, as the case may be. The routine then branches to the error routine (Chart CD). If no error nor end-of-file conditions are found, a branch is made back to this routine from RESTOR+4 to FIXIP1.

FIXIP1, EC-B3

The end-of-block address is stored in AEND (or BEND, etc.). When variable-length records are being processed, the input area address in FILEA (or FILEB, etc.) is incremented by four bytes for the block-length indicator. Then the file designation (A, B, C, or D) is restored in XFILE to be used in case a sequence error is detected later in the output routine. This routine then continues to TR1.

TR1, EC-D3

The input area address in register FILEA (or FILEB, etc.) is loaded in register MREG. Then, if so initialized, a branch-and-link is made to user exit 42 before continuing to LRA.

LRA, EC-F3

If the data is in a form that must be converted before merging, a 'T' (hexadecimal 'E3') is placed in location XTIME and a branch-and-link is made to STR3 for the initialized conversion routine. After linking back, this routine exits to the compare loops at an address that varies with the file in process and with the order of merge:

<u>File</u>	<u>Order of Merge</u>	<u>Branch to Address</u>	<u>Which is in Register</u>
A	4	COMP41	SAVEA
	3	COMP31	SAVEA
	2	COMP21	SAVEA
B	4	COMP42	SAVEB
	3	COMP32	SAVEB
C	4	COMP43	-----
D	4	COMP43	-----

GET1, EC-B1

This location is entered from the open/close routine whenever a new disk or tape volume was opened and from the compare loops whenever a block in a particular input area was depleted. The entry label will be GET1, GET2, GET3, or GET4.

The starting address of the input area for the current file is loaded in the

corresponding register (FILEA, FILEB, etc.) and in the corresponding CCW (for tape or disk). The routine then continues to TAPE1-2.

ERROR ROUTINE - ED

Tests are made for checkpoint record (tape only), end-of-volume and end-of-file, no record found (disk only), wrong-length record, and good, short block. All conditions except end-of-file are processed according to the condition found, after which a branch is made back to the input routine (Chart EC) to read the next block. For end-of-file, the depleted file is closed and merging is resumed with the remaining files. When all input files are closed, a branch is made to the end-of-job routine (Chart EJ).

In case of no record found, the remaining portion of the track is bypassed. In case of a data check, a bypass option and user exit 45 are available to permit the user to continue processing the record if so desired. In case of a wrong-length record the record will always be bypassed.

For valid records, the routine returns to FIXIP1 in the input routine to continue processing.

TEST, ED-A1

A WAIT macro is issued to ensure the read operation is complete before the routine continues to test for errors. Then, if the input file from which the block was just read is on tape, a branch is made to TAPEX.

For disk input, if a no-record-found condition is detected an end-of-track indicator (FF) is set in MREG+19 (ARADDR+3 or BRADDR+3, etc.) and a branch is made to TAPEG1-4 (or TAPEG2-4, etc.) to read the first block on the next track. Next, a test is made to determine if the block just read was the last block on the track. This test compares the next record count with the current one and if the next count is equal or lower, the end of the track has been reached. If such is the case, the end-of-track indicator (FF) is inserted in ARADDR+3 or BRADDR+3, etc. (MREG+19). Then, in either case, the data length is loaded in register RG15 and the routine continues to TMEOF.

TAPEX, ED-A2

The block just read is tested to determine if it was a checkpoint header record and, if so, a branch is made back to the input routine (Chart EC) to TAPE1 (or TAPE2, etc) to read the next block until a checkpoint trailer record is read, thus bypassing the checkpoint records. If not a checkpoint record, the block count is increased by one and the input block length is loaded in register RG15.

The CCB is tested for the WLR (wrong-length record) bit and if it is not on, a branch is made to TMEOF. If the CCB has a WLR indication, the residual count in the CCB is tested; if it is zero, a long WLR has been read and a branch is made to LALINK. If the residual count is not zero, it is subtracted from the input block length (in RG15) to obtain the length of the record read. A branch is then made to TMEOF.

TMEOF, ED-F1

The CCB is tested for end-of-file condition; if yes, a branch is made to EOFDK. If not end-of-file, the LINK register is loaded with the return address BPASS for use in the event that the previous block (fixed blocks only) was found to be a short block (BCERRW will be on). The routine then continues to BCERRW.

EOFDK, ED-G1

This location is entered when:

- End-of-file is detected at TMEOF.
- Initial open of input files is being performed.
- A good, short block at the end of a volume or a file is being processed.

An 'F' for end-of-file is placed in register RG12 (this will be stored in XFILE in the open/close routine and is significant for disk input only). The branch to EOFDK from the input routine is made a no-op and the routine continues to EOF.

EOF, ED-J1

This location is a branch to CEVCHK to call in the open/close routine at initial open time, at EOF or EOF for tape input, and at EOF for disk input. The return from the open/close routine is to open the next file at initial open, to close the particular file at EOF for tape or disk input, or to perform the next read operation at EOF for tape input.

At all other times, this location is a detected at the end of a volume or of a file. The branch to EOFDK from the input routine is activated so that when the records in the short block have been processed the input routine will branch to EOFDK. This routine then continues to FROMDK.

FROMDK, ED-C5

This location is entered from the preceding function at EOF or from CPIADD in the input routine, in both cases when a short block is being processed. The branch at EOF is activated and BCERRW is made a no-op. The routine then branches to RESTOR to process the short block as a valid record.

BCERRW, ED-B3

This location is a no-op until a short block is detected (for fixed block input only), at which time it is made an unconditional branch to ERRW.

Note: This location is entered only when EOF condition has not been detected. Then, if applicable, the WLR message is printed and the program returns to BPASS.

BPASS, ED-C3

This location is an unconditional branch to WLR if the BYPASS option was not specified. With the BYPASS option, a test is made for a data check in the CCB; if present, a branch is made to ERR. If there is no data check, the routine continues to WLR.

ERR, ED-E3

This location is an unconditional branch to ERROR if user exit 45 was not specified. Otherwise, a branch-and-link is made to the user's routine after which control is returned at one of two points:

- User desires to ignore error indication -- return (via B,14) to instruction preceding ERROR which is a branch to WLR to continue processing the block.
- User desires to bypass the block -- return (via B,4(14)) to ERROR.

ERROR, ED-H3

The count of unreadable records (ERRCT) is increased by one (ERRCT will be printed out at end-of-job) and the routine branches back to the input routine (Chart EC) to TAPEG1-4 (or TAPEG2-4, etc.).

WLR, ED-B4

If there is no key to read, this location is a branch to WLR2. With key, the key length is added to the data length (in RG15) before continuing to WLR2.

Note: In case of data check and no BYPASS option or no exit 45, IOCS does not return control to this program if input is on disk. For tape input, control is returned if the operator response to the IOCS message is 'ignore'. If BYPASS only or if exit 45 was specified, control is returned as described under ERR.

WLR2, ED-D4

Calculations and tests are made to determine if there is a wrong-length-record condition. This provision is made to prevent erroneous WLR indications for:

- variable-length records
- fixed-length records in variable blocks
- short, good blocks at the end of a volume or file of fixed-block, fixed-length records.

Register RG15 is saved so that its contents may be retrieved later (at RESTOR)

if required. For tape, this is the actual input block length and for disk it is the data length (plus the key length, if so specified). This value is compared to BLKINP which contains the input block length (for fixed blocks) or to the block-length indicator, the address of which is in RG10 (for variable blocks). If the result of the comparison is equal, it indicates a good record and a branch is made to RESTOR+4. If the result is low, it is a wrong length record for all types except fixed-length without key; a branch is then made depending on the type of record:

- Fixed-length without key - to MULTPL
- Fixed-length with key - to LALINK
- Variable-length - to LALINK

Note: For disk input, a WLR cannot be detected by testing the WLR bit in the CCB. Setting the SILI bit off in the CCW would cause a break in the disk CCW chain if a WLR condition is detected.

MULTPL, ED-G4

If the block length is not a multiple of the record length, a true WLR exists and a branch is made to LALINK. Otherwise, the branch is to LASTBLK (for fixed blocks) or to RESTOR (for variable blocks).

LASTBLK, ED-J4

The return branch at EOF is made a no-op and the no-op at BCERRW is changed to an unconditional branch to ERRW. The routine then branches with register RG13 back to the input routine (Chart EC) to TAPEG1-4 (or TAPEG2-4, etc.).

RESTOR, ED-D5

The contents of register RG15 are restored. For tape, this is the actual input block length and for disk it is the data length (plus the key length, if so specified). Then, at RESTOR+4, the end-of-block address is calculated by adding the contents of RG15 to the current address of the input area for the file in process (FILEA, FILEB, etc.) which is in register RG10.

The routine then branches back to the input routine (Chart EC) to FIXIP1 to continue processing the good block of records.

LALINK, ED-G5

The return address in the input routine (TAPEG1-4, TAPEG2-4, etc.) is loaded in the LINK register and the routine branches to ERRW.

ERRW, ED-H5

This location is entered from BCERRW or from LALINK; in either case, the function here is to print wrong-length-record message WIRMES, which contains the file designation (A, B, C, or D) retrieved from XFILE. The branch at BCERRW is made a no-op and EOF is made an unconditional branch to CEVCHK.

The routine then branches back to the address in the LINK register, the difference being due to the location from which ERRW was entered:

- From BCERRW, branch back to BPASS (previous block was a wrong-length record).
- From LALINK, branch back to the input routine (Chart EC) to TAPEG1-4 or TAPEG2-4, etc. (current block is a wrong-length record).

FILE D COMPARE LOOP - EE

For a 4-way merge, the initialization routine prepared certain branch instructions in compare loops D through B and the initial entry for merging is to this loop at COMP43. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the files being processed. As each file is depleted, an exit is made to the compare loop for the next lower order of merge (Charts EF and/or EG). A branch-and-link to the output routine (Chart EH) moves each winning record, in turn, to the output area for further processing. The program keeps returning to this loop as long as there are records to process in file D. It then exits to the file C compare loop (Chart EF).

COMP43, EE-B3

The exit from this loop is provided at the beginning so that no processing need be done when file D is depleted. If such is the case, a branch is made to COMP32 in the file C compare loop (Chart EF); if not, a record from the file D sequence is compared with a record from the file C sequence. As long as D is determined to be the winner, it is compared with records from the other available sequences, in turn. The routine thus continues in this compare loop or exits to another loop, depending on the results of each comparison. For example, when merging in ascending sequence:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMP43	D	COMP42
	C	COMP32 (Chart EF)
COMP42	D	COMP41
	B	COMP21 (Chart EG)
COMP41	D	PUT4
	A	PUT1 (Chart EG)

The branch locations are determined not only by the results of the comparison but also by the depletion of input sequences. For example, if in COMP42 sequence B is found to be depleted, the instruction at B41 is an unconditional branch to COMP41 to compare the D record with the next A record.

Another variation in the compare loop operation occurs when, for example, a D record is found to be the winner in COMP43. Then, in COMP42, the B record is found to be the winner. The exit from the loop, as previously described, is to COMP21 (Chart EG); however, the return address that is saved in register SAVEB is COMP42+2. Then, assuming the B record is the winner in compare loop A and is moved to the output area, control is returned to the D loop at the address in SAVEB. The reason for entering this loop at COMP42+2 is that although B was the winner, the D record had already been determined to be winner over C at that time. Therefore, the comparing in D loop resumes at the point where D is compared with the next B record.

PUT4, EE-F3

The address of the winning record from the file D sequence is loaded into register MREG and a branch-and-link is made to OUTFIL in the output routine (Chart EH). Control is returned to this routine where the address for the file D input area is

updated. If there are no more records in the area, a branch is made to GET4 in the input routine (Chart EC) to refill the input area. If however, there are more records in the D input area, the compare loop is re-entered at COMP43 via TR4-4 in the input routine.

FILE C COMPARE LOOP - EF

For a 3-way merge, the initialization routine prepared certain branch instructions in compare loops C and B and the initial entry for merging is to this loop at COMP32. However, the flow through these loops varies not only with the order of merge but also, later on, with the depletion of records in the files being merged. As each file is depleted, a branch is made to the compare loop for the next lower order of merge (Chart EG). A branch-and-link to the output routine (Chart EH) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from file C. It then exits to the file B compare loop (Chart EG).

COMP32, EF-B3

The exit from this loop is provided at the beginning so that no processing need be done when file C is depleted. If such is the case, a branch is made to COMP21 in the file B compare loop (Chart EG); if not, a record from the file C sequence is compared with a record from the file B sequence. If the C record is the winner, it is compared with a record from sequence A. If the C record wins again, the routine continues to PUT3.

The other branch locations, in the event that either B or A is determined to be the winner, are:

<u>Function</u>	<u>Winner</u>	<u>Branch to</u>
COMP32	B	COMP21 (Chart EG)
COMP31	A	PUT1 (Chart EG)

The branch locations are determined not only by the results of the comparison but also by the depletion of input sequences. For example, if in COMP31 sequence A is found to be depleted, the instruction at BPUT3 is an unconditional branch to PUT3 to prepare to move the C record to the output area.

Another variation in the compare loop occurs when, for example, a C record is found to be the winner in COMP32. Then, in COMP31, the A record is found to be the winner. The exit from the loop, as previously described, is to PUT1 (Chart EG); however, the return address that is saved in register SAVEA is COMP31+2. Then, after the A record is moved to the output area, control is returned to the C loop at the address in SAVEA. The reason for entering this loop at COMP31+2 is that although A was the winner, the C record had already been determined to be winner over the B record. Therefore, the comparing in C loop resumes at the point where C is compared with the next A record.

PUT3, EF-E3

The address of the winning record from the file C sequence is loaded into register MREG and a branch-and-link is made to OUTFIL in the output routine (Chart EH). Control is returned to this routine where the address for the file C input area is updated. If there are no more records in the area, a branch is made to GET3 in the input routine (Chart EC) to refill the input area. If, however, there are more records in the C input area, the compare loop is re-entered at COMP43 via TR3-4 in the input routine.

FILE B COMPARE LOOP - EG

For a 2-way merge, the initialization routine prepared certain branch instructions in compare loop B and the initial entry for merging is to this loop at COMP21. However, the flow through this loop varies not only with the order of merge but also, later on, with the depletion of records in file B. When file B is depleted, a branch is made to prepare to move the A record to the output area.

A branch-and-link to the output routine (Chart EH) moves each winning record, in turn, to the output area. The program keeps returning to this loop as long as there are records to be merged from file B. It then branches directly to PUT1.

COMP21, EG-C2

The exit from this loop is provided at the beginning so that no processing need be done when file B is depleted. If such is

the case, a branch is made to PUT1; if not, a record from the file B sequence is compared with a record from the file A sequence. If the B record is the winner, the routine branches to PUT2; if the A record is the winner, the routine branches to PUT1.

PUT2, EG-E2

The address of the winning record from the file B sequence is loaded into register MREG and a branch-and-link is made to OUTFIL in the output routine (Chart EH). Control is returned to this routine where the address for the file B input area is updated. If there are no more records in the area, a branch is made to GET2 in the input routine (Chart EC) to refill the input area.

If, however, there are more records in the B input area, the compare loop is re-entered at the address in register SAVEB via TR2-4 in the input routine. This address is COMP42 if the OM is 4, or COMP32 if the OM is 3.

PUT1, EG-E4

The address of the winning record from the file A sequence is loaded into register MREG and a branch-and-link is made to OUTFIL in the output routine (Chart EH). Control is returned to this routine where the address for the file A input area is updated. If there are no more records in the area, a branch is made to GET1 in the input routine (Chart EC) to refill the input area. If, however, there are more records in the A input area, the compare loop is re-entered at the address in register SAVEA via TR1-4 in the input routine. This address is COMP41 if the OM is 4, or COMP31 if the OM is 3, or COMP21 if the OM is 2.

OUTPUT ROUTINE - EH

When a winning record has been determined by the mainline compare loops, the output routine is entered at OUTFIL. This routine:

- Sequence checks the output file.
- Reconverts data, if specified.
- Exits to user routine, if specified (via exit 43).

- Updates the main storage output area.
 - Moves records to the main storage output area.
- Note: This can be either winning records or user-inserted records.
- Maintains a count of records merged.
 - Writes blocks of records on tape or disk whenever the output area fills up.
 - Provides for maximum usage of track capacity.
 - Tests for end-of-volume and end-of-job and branches to the appropriate routine for further processing.

The routine normally exits back to the compare loops to the address in the LINK register. For end-of-volume, the open/close routine is called in to process the condition. When end-of-job is detected, it exits to the end-of-job routine (Chart EJ).

OUTFIL, EH-B1

The address in register LINK is stored so that this register may be used to branch-and-link to user and/or reconversion routines, if specified.

The first record to be moved to the output area cannot be sequence-checked because there is no preceding record in the output area. Also, the user and reconversion routines, even if specified, cannot be executed for the first record because these functions are performed on records that already have been moved to the output area. Therefore, a branch is made to the instruction preceding FULOUT where the instruction at SEQCK1 is made a no-op so that all subsequent records will be sequence-checked.

Beginning with the second winning record, each record is compared with the one that preceded it to the output area. If the record to be moved is out of sequence, a branch is made to SEQERR; otherwise, the routine branches to CONVRT.

SEQERR, EH-D1

The address of the out-of-sequence record (in MREG) is saved in register RG13, the address for the return to this routine is

loaded in register MREG, and the file identification (A, B, C, or D in XFILE) is loaded in register RG12. A branch is then made to CEVCHK to call in overlay DSORT401 to process the error condition in the sequence-error routine (Chart EL). If the merge is to continue, the program returns to this routine at SEQBACK (via EXIT in DSORT401).

SEQBACK, EH-H1

The address of the out-of-sequence record is restored to register MREG and a branch is made to CONVRT to continue processing. The out-of-sequence record will thus be moved to its output location either because an IBM 1052 Printer-KeyBoard was not available or, if it was, because the user response was "ignore".

CONVRT, EH-J1

This location may have been initialized to branch to EXIT (if reconversion is not required) or to UPDTP (if neither reconversion nor exit 43 are required). However, if the data was converted for merging, it must now be reconverted to its original format. An 'R' (hexadecimal D9) is placed in XTIME and a branch-and-link is made to STR3 for the initialized reconversion routine. Control is returned to this routine at EXIT.

Note: If reconversion is required, it will be done after the sequence check and before the user exit.

EXIT, EH-A2

This location will have been initialized to branch to UPDTP after data reconversion if user exit 43 is not required; otherwise, a branch-and-link is made to the user's routine. Control is returned by the user either at UPDTP or at the instruction preceding UPDTP; in the latter case, the user did not insert a record so location INSERT is made a no-op before continuing to UPDTP.

Note: The record made available to the user at exit 43 is the one that preceded the current winning record and that has been sequence-checked and, if necessary, reconverted.

The output area (FILEO) address is increased by the value in OUTLGN, which is:

- For fixed-length records, the length of the output record (L3, which was moved to OUTLGN during initialization).
- For variable-length records, the length of the last record moved.

When variable-length records are being processed, the program provides for maximum usage of each track's capacity. The end address of the output area is carried in two locations:

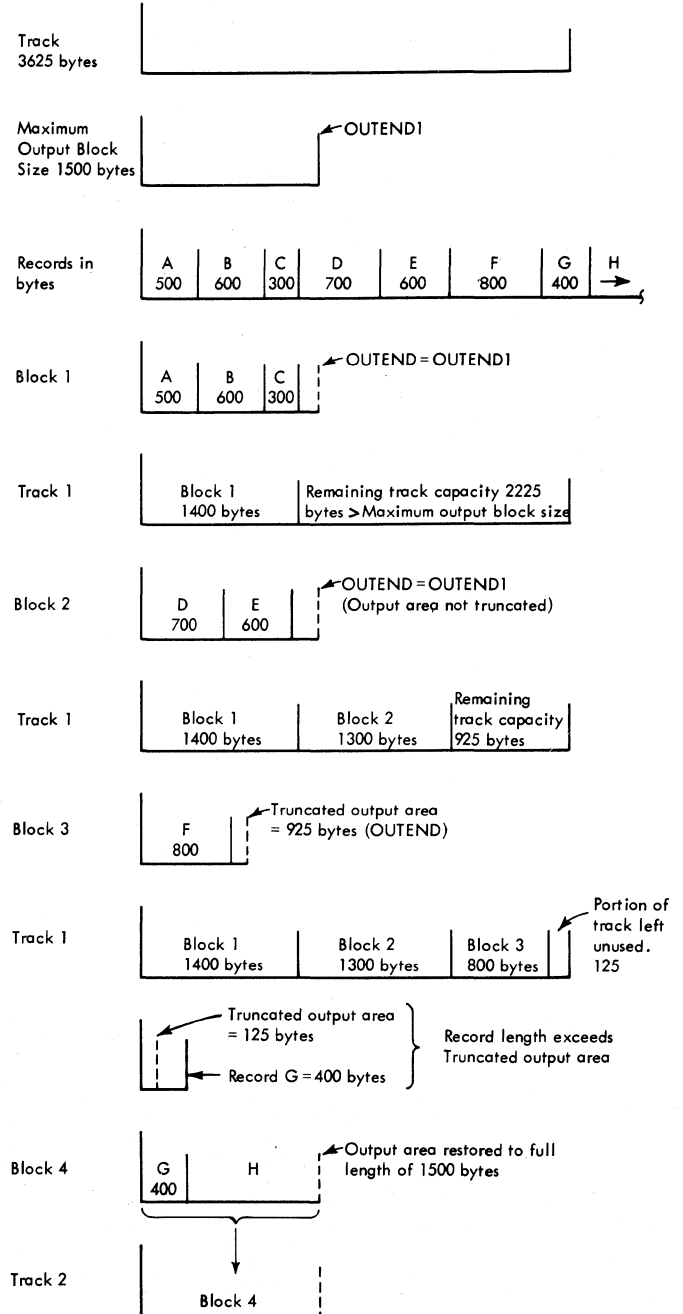
- OUTEND1, which contains the original end address throughout the merge phase.
- OUTEND, which will contain the original end address (OUTEND1) at least until after the first block has been written and the amount of track capacity remaining has been calculated. If this amount is less than the maximum output block size, the output area is truncated so that it equals the track capacity remaining and the new end address is inserted in OUTEND for testing the next record.

After the first block of a track has been built in the main storage output area, the track capacity remaining after it is written on disk is calculated. If the track capacity remaining is equal to or greater than the maximum output block size, the block is written and the next record is unconditionally moved to the output area.

If the track capacity remaining is less than the maximum block size, the main storage output area is truncated (shortened) so that it equals the remaining track capacity, and the new end address of the main storage output area is placed in OUTEND. The block is written on disk and the next record is unconditionally moved to the main storage output area. A test is made to determine if the record unconditionally moved exceeds the shortened output area. If it does not, the record length indicator of the next record to be moved to the main storage output area is tested to determine if the record will also fit in the output area, and the next output block is built as described in FULOUT. If the unconditionally moved record does exceed the shortened output area, the output area will be restored to its original length by moving the value of OUTEND1 into OUTEND. The new output block, the first record of which is the unconditionally moved record, is built and

written on the next available track. (See Figure 49.)

The branch at SEQCK1 is made a no-op, after the first record is moved to the main storage output area, to permit sequence-checking the next record; the routine then continues to FULOUT.



Note: Block sizes in bytes are given for illustration only and do not represent the actual number of bytes consumed in writing the equivalent block.

Figure 49. Maximum Use of Output Track Capacity

FULOUT, EH-F2

A test is made to determine if the output area is full. If it is, a branch is made to LWRITE for fixed-length records or to WRITEV for variable-length records.

If the record will fit in the output area, one of two courses of action is taken:

- For fixed-length records, the record length (FIXDRL if L1>L3 or OUTLGN if L1<L3) is loaded in a register, reduced by one, and a branch is made to FIXMOV.
- For variable-length records, the length of the record to be moved is obtained from register MREG (via OUTRLI+2) and stored in OUTLGN. The length is reduced by one and the routine continues to FIXMOV.

FIXMOV, EH-J2

The record length, minus one, is stored in OUTRLI+2 and registers RG4 and RG5 are initialized for moving 256 bytes at a time to the output area. Register FILEO (output area address) is saved in case user exit is made later (at INSERT) because FILEO will change if there are more than 256 bytes to move.

Records are moved from the address in MREG to that in FILEO, using the contents of registers RG4 and RG5 in a loop that moves 256 bytes at a time. MREG and FILEO are updated each time through the loop. When the count in index register RG1 becomes greater than that in RG5, the length of the last move is calculated and the remaining bytes are moved to the output area.

Example: 600-byte record to be moved.
From FULOUT, register RG4=256 and register RG5=600.

	<u>RG4</u>	<u>RG5</u>	<u>RG1</u> (Index)
Start	256	600	0
1st move	256	600	256
2nd move	256	600	512
3rd move*	256	600	768

* (Not executed because RG1 is neither lower nor equal to RG5)

Then, register 4 plus register 5 minus register 1 = 88 remaining bytes.

The final move thus consists of 88 bytes and is executed by the instructions at RESTMV. After the entire record has been moved, the record length is incremented by one to restore the original value. MREG is updated to the next record location and registers FILEO, RG4, and RG5 are restored.

INSERT, EH-K2

This location is a branch to ZYXWZY if user exit 43 was not specified. When exit 43 is initialized, if the last record moved is a user-inserted record, a branch is made to EXIT to permit multiple insertions by the user. Otherwise, the routine continues to ZYXWZY.

ZYXWZY, EH-J3

After each record is moved to the output area, the count in RCOUNT (number of records processed) is increased by one and the new total is stored back in RCOUNT. The LINK register is restored and the routine branches back to the compare loop in which the winning record was found to test if the block in that input area is depleted.

WRITEV, EH-B3

When variable-length records are to be written on disk or tape, OBEGIN (the starting address of the output area) is incremented by eight to bypass the record count at the beginning of the output area. The block-length indicator is then calculated and stored following the record count.

For tape output, the data count is stored in the write CCW and a branch is made to LWRITE. For disk output, the output block length is stored in the data-length portion of the disk count and in a work register. The data count is incremented by 8 and stored in the write CCW and the verify CCW. Calculations are then made to determine if the records in the output area will fit on the current track; if not, the track capacity is restored to 3625 and the end-of-track indicator (FF) is set. The area remaining on the track is then calculated and stored in TKLEFT.

A calculation and test are now made to determine if the amount of track left is

less than the maximum output block size. If so, the output area is truncated to equal the amount of track left (for the reason described in the text under UPDTC). This lower value for the end address of the output area is stored in OUTEND. If the amount of track left is equal to or greater than the maximum output block size, the original end address of the output area is left in OUTEND. Whenever the output area is truncated, it will be restored to its original length at UPDTC if the record moved does not fit as the first record in the truncated area, after the last block is written on the current track.

The routine then continues to LWRITE.

LWRITE, EH-A4

OBEGIN (the starting address of the output area) is restored in register FILEO. For tape output, a branch is made to TAPPUT.

For disk output, the current disk address of the output file, ORADDR, is incremented by one. Then, in the case of fixed-length blocks, a test is made to determine if the record number has exceeded the maximum on the current track. If it has, the 256-complement of the blocks-per-track count is added to obtain a new head number. In the case of variable-length blocks, a test is made for the end-of-track indicator. If present, the record number is made one. Then, for both variable- and fixed-length blocks, the maximum head number is tested. If exceeded, a new cylinder number is calculated. Finally, the output extent upper limit is tested. If not exceeded, the new cylinder number is stored in ORADDR, and register RG0 is decreased by one and stored in CHHR for the seek command in the write CCW chain. The routine then continues to TAPPUT. If the limits are exceeded, the branch-back address for disk (TAPEO) is loaded in register RG13 and a branch is made to EOVDISK.

TAPPUT, EH-C5

An EXCP macro is issued and IOCS writes a block of records on the disk or tape output file. The parameters for the write operation are supplied by OCCB. When control is returned by IOCS, this routine branches to ADJOUT if the output is on disk; if on tape, the output block count is updated before continuing to ADJOUT.

ADJOUT, EH-F5

The address in register FILEO is increased by eight bytes for fixed-length records, or twelve bytes for variable-length records, to obtain the address of the first byte of data. This increase bypasses the count field and, in the case of variable-length records, the BLI as well.

A test is then made for end-of-volume and, if detected, the branch-back address for tape (PH3EJ1) is loaded in register RG13 and the routine continues to EOVDISK. If not end-of-volume, the routine continues to PH3EJ1.

PH3EJ1, EH-H5

The functions performed at this location vary with the type of record and end-of-job conditions:

- If end-of-job, a branch is made to the end-of-job routine (Chart EJ) at PH3EJ2 (after the last block has been written) or at TAPOUT (after the EOF indicator has been written).
- If not end-of-job, a branch is made back to FULOUT+8 for fixed-length records or to VARMOV for variable-length records to prepare to move the record to the output area (the record that would not fit previously because the output area was full).

Note: This is the only time that a record will be moved without first testing to determine if it will fit in the output area. It is certain to fit because it is the first one after a block has been written. (See note at end of UPDTC.)

EOVDISK, EH-G4

Register MREG is saved and the address of OCCB is loaded in MREG. A 'V' is placed in register RG12 and the open/close routine (Chart EM) is called in (via CEVCHK) to process the end-of-volume condition.

If end-of-job has not yet been reached, control is returned to this routine (via CEVCHK). After restoring register MREG, a branch is made to the address in register RG13, which is FULOUT+8 for fixed-length records on tape, or VARMOV for variable-length records on tape, or TAPEO for either record type on disk.

END-OF-JOB ROUTINE - EJ

This routine is entered when all the input files have been closed. Its principal functions are:

- Initialize to reconvert the last record moved and to exit to the user (if required).
- Initialize to write the last block.
- Initialize to write an EOF record (disk output only).
- Insert an EOF indicator for the open/close routine.
- Call in the open/close routine to close the output file, print an EOJ message, and end the job.

PH3EOJ, EJ-B3

Register RG10 is made negative to indicate to the user that the last record is in the output area. The instruction before UPDTC and the one at ENTRY (both in the output routine, Chart EH) are modified to branch back to this routine at ADD1 after the last record has been processed. A branch is then made to CONVRT in the output routine. If necessary, the record is reconverted and a branch is made back to ADD1 if user exit 43 was not specified; otherwise, the output routine exits to the user routine after which a branch is made back to ADD1.

ADD1, EJ-E3

The output routine is modified to write the last block from the output area and a branch-and-link is made to WRITEV (if no key) or to LWRITE (with key) in the output routine. After the last block is written, a branch is made back to this routine at PH3EJ2 from the instruction following PH3EJ1 in the output routine.

PH3EJ2, EJ-H3

For tape output, a branch is made to TAPOUT. For disk output, OBEGIN (address of start of output area) is loaded into register FILEO, eight is added to FILEO, and a branch-and-link is made to WRITEV in the output routine to write an end-of-file record on the disk. At WRITEV, register

RG1, which contains OBEGIN+8, is subtracted from FILEO, thus giving a result of 0 for the data length. This data length of 0 on disk is the end-of-file record. A branch is then made back to this routine at TAPOUT from the instruction following PH3EJ1 in the output routine.

TAPOUT, EJ-J3

The output CCB (OCCB) is loaded in register MREG and an 'F' for end-of-file is loaded in register RG12. The open/close routine is then called in (via CEVCHK) to close the output file and to branch to the end-of-job messages routine. The job will then be ended with an EOJ macro.

END-OF-JOB MESSAGES - EK

After the last block has been written out in the end-of-job routine (Chart EJ) in the mainline, overlay DSORT401 is called into main storage and this routine is entered through CLOSEO in the open/close routine (Chart EM).

EOJMES, EK-C3

The number of unreadable blocks that were bypassed in the mainline is obtained from location ERRCT. If the number is zero, a branch is made to PH4EOJ; if not, message BYPMES is printed:

```
'7DD4I PHASE 4 UNREADABLE BLOCKS  
BYPASSED 0000'
```

The zeros at the end of the message are replaced by the number from ERRCT.

PH4EOJ, EK-E3

The number of records processed by phase 4 is obtained from location RCOUNT and messages RECMES and MESEJO are printed:

```
.'7DD5I RECORDS PROCESSED 000000000'
```

```
'7DD6I END OF MERGE'
```

The zeros at the end of the first message are replaced by the number from RCOUNT.

The EOJ macro is then issued and the job is terminated.

SEQUENCE-ERROR ROUTINE - EL

When a record is found to be out of sequence in the output routine (Chart EH) in the mainline, overlay DSORT401 is called into main storage and this routine is entered through COMP in the open/close routine (Chart EM). The branch to this routine is made when COMP detects that the mainline return address is lower than that of the CCB for file A.

The routine first prints a sequence-error message. Then, if a 1052 Printer-KeyBoard is provided, it checks for the operator response and either returns to the output routine in the mainline to resume processing or cancels the job. If a 1052 is not provided, the routine returns to the output routine immediately after printing the message.

SEQERR, EL-C3

The file designation (A, B, C, or D) is obtained from XFILE and inserted in the error message (SEQMES1). An EXCP macro is issued and the message is printed:

'7DD2D SEQ. ERROR FILE n'

The routine then continues to RESPONSE.

RESPONSE, EL-E3

When this overlay was initialized (Chart EA), the instruction at this location was made an unconditional branch to EXIT (Chart EM) if the system does not include a 1052 Printer-KeyBoard. In this case, the program returns to the output routine at location SEQBACK (Chart EH) and processing is resumed.

When a 1052 is provided, the operator response appears in SEQMES2 and an EXCP macro is issued to read the response. Tests are made for the words 'IGNORE' and 'CANCEL' in both upper- and lower-case characters and one of three courses of action is taken, depending on the results:

- IGNORE or ignore - branch to EXIT to return to output routine at SEQBACK (Chart EH).
- CANCEL or cancel - Issue CANCEL macro.
- None of preceding - Issue EXCP macro and print message SEQMES3:

'7DD2A INVALID RESPONSE'.

When an invalid response is made, the routine branches back to read the corrected response as described in the preceding paragraph.

OPEN/CLOSE ROUTINE - EM

This routine is read from disk into main storage whenever the merge mainline detects an end-of-volume or end-of-file condition. It is also called in at the start of the merge phase to perform the initial OPEN for the output file and all the input files.

The routine was initialized (Chart EA) for the type of output file and of each input file (disk, tape, or mixed) as well as for label types and user options. Its functions now vary not only with the manner in which it was initialized, but also with the condition for which it was called and with conditions that are detected within the routine and in IOCS.

The major functions are:

- Open the output file and all the input files required by the order of merge. This is done at initial open time.
- Provide linkage to IOCS and between IOCS and the user's routines at exits 41 and 44. Exit linkage is provided only when specified.

Note: When exit 41 is specified by the user, the sort/merge program turns on the corresponding bit in the DTF tables for all the input files. IOCS will then provide linkage to exit 41 only for those files that have non-standard or user labels.

- Open and close individual input volumes and files as required by the merge mainline, IOCS, or this routine.
- Open and close output volumes (if multivolume output) and, when the output file is closed, branch to the end-of-job routine (Chart EJ).
- Test if entry was for sequence error and, if so, branch to the sequence-error routine (Chart EL).

After each execution, the routine is written back on disk in the checkpoint track and the merge mainline resumes processing in all cases except at end-of-job or if job is canceled by user because of sequence error.

BEGIN, EM-B2

The contents of registers RG10 through RG15 are stored so that these six registers can be used in this routine.

The first time that this routine is called into main storage by the merge mainline will be when the output file is to be opened. The address of the output CCB (which is in register MREG) is stored in ADDROUT and the routine continues to OPENO where the output file (FILEO) is opened. If the user has specified exit 44, IOCS branches to USEREX44. When control is returned to this routine by IOCS, the program continues at NIEFO.

The second entry to this routine (after the output file is opened) branches directly to OPENA.

USEREX44, EM-B5

When the user has specified exit 44 in the MODS control card, IOCS branches back to this location after each OPEN and CLOSE of an output volume or file.

The address of the user's routine (USADDR) is loaded and a branch-and-link is made to the user's routine. Control is returned by the user at one of two locations where the appropriate LBRET macro is issued: LBRET1 indicating no more labels or LBRET2 indicating more labels. Control is then returned to IOCS.

NIEFO, EM-E2

For disk output, the upper and lower limits of the extent are stored at ORADDR (OCCB+16) and the logical unit address is stored in OCCB in the merge mainline; the routine then continues to EXIT.

For tape output, the routine branches directly to EXIT.

EXIT, EM-H2

The contents of registers RG10 through RG15 are restored and a branch is made back to the merge mainline to the address in register LINK.

OPENA, EM-B3

When this location is entered for the first time, the first input file (file A) is to be opened. The address of the CCB for file A (ACCB, which is in register MREG) is stored in ADDRINP. The instruction at BEGIN+4 is then made an unconditional branch to OPENX for all future entries during this job because the initial open of the output file and of the first input file (OM is at least 1) will not be repeated. The routine now continues to OPENX.

All subsequent entries to this routine will branch directly to OPENX.

OPENX, EM-D3

The end-of-volume ('V') or end-of-file ('F') indicator, which was inserted in register RG12 by the merge mainline, is stored in XFILE (this indicator will have no effect when input is on tape). Registers RG12 and RG13 are loaded with the addresses of OCCB and ACCB, respectively, and a test is made (using RG12) to determine if this routine was entered for the output file. If so, a branch is made to OUTPUT; for an input file, the routine continues to COMP.

COMP, EM-D4

A test is made (using RG13) for the input file that is to be opened or closed (from A to D, depending on the order of merge). If the result of the test is:

- lower than A - a sequence error was detected in the merge mainline; branch to sequence-error routine (Chart EL).
- equal to A - file A is to be opened or closed; branch to INPUT.
- higher than A - file B, C, or D is to be opened or closed; get the addresses of next highest DTF and CCB and repeat the test until an equal condition results; branch to INPUT.

OUTPUT, EM-F3

This location was initialized for either disk or tape output:

- Disk - Branch to OUTDK

- Tape - Move the output block count into the tape output DTF (FILEO+40), then reset the block count (in OCCB+20) to zero. Continue to OUTDK.

OUTDK, EM-H3

A test is made for end-of-file ('F' in XFILE). If such is the case, a branch is made to CLOSEO; if not, the branch depends on the type of file:

- Disk - to OPENO
- Tape - to FEOVOUT

FEOVOUT, EM-J5

An FEOV macro is issued, IOCS closes the tape output volume and opens the next volume. If user exit 44 was specified, IOCS branches back to USEREX44 in both cases (close and open). When control is returned to this routine by IOCS, the FEOV bit in the DTF table is turned off and a branch is made to EXIT.

CLOSEO, EM-J3

A CLOSE macro is issued and IOCS closes the output file. This occurs after the last block has been written and an 'F' has been placed in register RG12 in the merge mainline. If user exit 44 was specified, IOCS branches back to USEREX44. When control is returned to this routine by IOCS, the program continues to the end-of-job routine (Chart EJ).

INPUT, EM-E4

The processing between this point and GOGO1 varies with the OM (order of merge) and the time of entry to INPUT. The purpose of this portion of the routine is to first perform the initial OPEN for all the input files required by the OM, then to prepare for subsequent entries, and finally to bypass this portion when it is no longer needed. The several courses of action are:

1. On the first entry to INPUT, at least one input file must be opened, therefore:
 - a. As long as the OM is one or more,

there are additional input files to open. The OM is decremented by one, a 'V' is placed in XFILE (to denote not end-of-file), and a branch is made to GOGO1.

- b. When the OM has been decremented to zero, all but one of the required input files have been opened. A branch is activated from INPUT to GOGO, a 'V' is placed in XFILE, and a branch is made to GOGO1 to continue processing for the initial OPEN of the last file.

2. When INPUT is entered the first time after the initial OPEN's have all been executed, a branch is made to GOGO. Here, the instruction at location MODIFY is initialized to return to the mainline at the appropriate GETn instruction (see MODIFY for further details). Then a branch is activated from INPUT to GOGO1 and the routine continues to GOGO1.
3. All subsequent entries to INPUT branch directly to GOGO1 because the functions described in the preceding items 1a, 1b, and 2 are no longer needed.

GOGO1, EN-C1

The address of the DTF table for the file to be opened or closed is stored in the OPEN and CLOSE macros. The file designation (A, B, C, or D) is stored in FILETYPE+3 to be available to the user (in case exit 41 is specified).

A test is then made for tape input and, if positive, a branch is made to TAPEIN. For disk input, the switch at TPKSW is turned off (no-op) and a 'D' is stored in FILETYPE+1 to be available to the user (in case exit 41 is specified). Location XFILE is then tested and if a 'V' is detected a branch is made to OPENI. If an 'F' is detected in XFILE, the unit exception bit is turned on in the CCB portion of the DTF table for the current file and the routine continues to OPENI.

OPENI, EN-G2

An OPEN macro is issued and IOCS opens the designated input file (or volume of a file). If the user has specified exit 41, IOCS branches to USEREX41. When control is returned to this routine, one of two courses of action is taken at TPKSW:

- For tape (TPDKSW is ON) - Branch to NIEFI
- For disk (TPDKSW is OFF) - The upper and lower limits of the extent are stored at ARADDR in the merge mainline (or BRADDR, etc., as the case may be). The logical unit address is stored in the CCB for the file that was just opened and the routine continues to MODIFY.

MODIFY, EN-J3

This instruction was initialized earlier in this routine (see text for item 2 under INPUT) after the initial OPEN of all the required input files. It is entered after each subsequent OPEN of disk input files, except when an end-of-file is detected.

The address in the mainline to which the program will return (after this open/close routine is written back on the checkpoint track) is incremented so that the branch to the input routine will be to the next GET_n instead of to the USTOP_n for that file (Chart EC). After incrementing register MREG by the required amount, a branch is made to EXIT.

NIEFI, EN-J2

The instruction at this location was initialized for the CLOSE rewind option. The bit in the DTF table (for the file that was just opened) is now set for the specified option and a branch is made to EXIT.

USEREX41, EN-B3

When the user has specified exit 41 in the MODS control card, IOCS branches back to this location after each OPEN and CLOSE of an input volume or file*. After designating the type of labels and the file in process, the address of the user's routine (USADDR) is loaded and a branch-and-link is made to the user's routine. When control is returned by the user, an LBRET2 macro is issued by this routine and control is returned to IOCS.

 *See note in the introduction to this routine pertaining to the linkage between IOCS and exit 41.

TAPEIN, EN-E1

The switch at TPDKSW is turned on (unconditional branch) and a 'T' is stored in FILETYPE+1 to be available to the user (in case exit 41 is specified). The routine then continues to SWITCH2.

SWITCH2, EN-F1

As long as the order of merge for tape (TPOM) is one or more, the routine branches to OPENI for the initial OPEN of each tape input file. When TPOM has been decremented to zero, SWITCH2 is made in unconditional branch to GOGO2 for end-of-volume or end-of-file processing.

GOGO2, EN-G1

The input block count is reduced by one because it would have been incremented in the mainline before the EOVS or EOF was detected. The corrected count is stored in the DTF table for the current file and is then reset to zero for the next volume or file.

If the last volume of a file is to be closed (test VOLA, which is equal to current CCB+19), a branch is made to EOFTAPE. If there are more volumes in the current file, the label type is obtained from LABTYP_n (current CCB+16) and used to initialize the branch at FEOVTP:

- Standard labels and non-standard labels with user exit - activate the branch to MEND.
- Non-standard labels without user exit and unlabeled files - do not activate branch.

The number of volumes in the current file is reduced by one and the routine continues to FEOVTP.

FEOVTP, EN-H5

An FEOV macro is issued, IOCS closes the tape input volume and opens the next volume. When control is returned to this routine, the FEOV bit in the DTF table is turned off.

The address in the mainline to which the program will return (after this open/close

routine is written back on the checkpoint track) is incremented at location ADD26 so that the branch to the input routine will be to the next GET_n instead of to the USTOP_n for that file. After incrementing register MREG by the required amount, a branch is made to EXIT.

Note that FEOVTP is entered for unlabeled files and for non-standard labels without user exit. In both cases, the user exit bit will be off.

EOFTAPE, EN-G4

The bit in the DTF table at FILE_n+32 is set on to force an end-of-file condition, and the routine continues to MEND.

MEND, EN-H4

The FEOV bit is turned off in the DTF table for the current input file, and an expansion of the END macro is executed. If user exit 41 was specified, IOCS branches back to USEREX41.

If IOCS detects an end-of-file condition (either in the EOF bit which was turned on in the DTF by this program or in labels), it branches back to CLOSEI, which is the end-of-file address that IOCS finds in the DTF table. If end of file is not detected, when IOCS returns control to this routine the branch address in the mainline is modified at location ADD26 so that the branch to the input routine will be to the next GET_n instead of to the USTOP_n for that file. The routine then branches to EXIT.

CLOSEI, EN-B4

A CLOSE macro is issued and IOCS closes the input file. If user exit 41 was specified, IOCS branches back to USEREX41. When control is returned to this routine by IOCS, a branch is made to EXIT.

Note that when a CLOSE macro is executed, the address in the mainline to which the program returns will not have been incremented at MODIFY nor at ADD26. The return point is therefore to USTOP_n (for the file that was just closed) and the compare loop for that file is deactivated or closed.

CHECKPOINT ROUTINE - EP

This routine is used to alternately write on disk and read into main storage overlays DSORT401 and DSORT402. At the beginning of phase 4, DSORT401 is loaded by job control and, at the completion of its initialization, it is written on disk and DSORT402 is fetched by a self-contained subroutine in DSORT401. All subsequent linkage between the two overlays is made by this checkpoint routine which is in the resident portion of DSORT402 in main storage.

The read and write functions are performed by the same set of instructions, starting at location SVCKPT. The first instruction changes the command code in the read/write CCW from 'read data' (06) to 'write data' (05) alternately each time it is executed, the second one loads the CCB address (CHKCCB), and the third issues the EXCP macro. The program is assembled with the read command code in the CCW and it is left that same way at the end of each execution of this routine. The routine is written so that the checkpoint track number is also alternated as required: Track 1 for DSORT402, Track 2 for DSORT401.

When this routine is entered at CEVCHK for initial open of files or for EOF or EOF, register RG9 (MREG) will have been loaded with the address of the CCB for the file in process. The return linkage to the mainline is then made with MREG after the DSORT401 functions are completed. Note that each CCB in the mainline is followed by three full words and one halfword which contain constants that are used throughout the phase. Not all of these constants are used by the DSORT401 overlay, but they were placed at these locations to maintain the same displacement between the CCB's. For the same reason, no-ops have been placed in the series of instructions following USTOP3 and USTOP4. Thus, USTOP1 through USTOP4 are located at MREG+30 for ACCB through DCCB, and the branches to GET1 through GET4 are at MREG+30+26 (the 26 additional bytes are added by MODIFY for disk or by ADD26 for tape in the open/close routine, Chart EN, and are used for return after end of volume).

The various linkages between overlays and within overlay DSORT401 are summarized in Figure 50.

Enter from	Routine	Purpose	Return to
FILLDL	Initialization	Open output file	OCCB+30, which is a branch via register RG13 to TAPE1-2 in the input routine
EOFDK	Error	<ol style="list-style-type: none"> 1. Initial open of input files 2. EOF on input tape (open next volume) 3. EOF on input tape or disk 	<ol style="list-style-type: none"> 1. nCCB+30, which is USTOP1 through USTOP4, per order of merge 2. nCCB+30+26, which is GETn for corresponding file 3. nCCB+30, which is USTOPn for corresponding file (to close loop for depleted file)
CPIADD	Input	EOV on input disk (get next extent)	nCCB+30+26, which is GETn for corresponding file
SEQERR	Output	Sequence error	MREG+30, which is SEQBACK in output routine, if the sequence error is to be ignored; if not, see Chart EL.
EOVDSK	Output	<ol style="list-style-type: none"> 1. EOV on output tape or disk 2. EOF on output tape or disk 	<ol style="list-style-type: none"> 1. OCCB+30, which is a branch via register RG13 to TAPEO (for disk) or PH3EJ1 (for tape) in the output routine 2. EOJMES, Chart EK

Figure 50. Linkages Between Overlays

At USTOPn (MREG+30), when the input files are being opened, the compare loop branches are initialized for the corresponding file and a bit is set in the merge-status indicator (PH2MOM). The instruction at USTOPn+20 is a no-op at this time so that a branch is made to GETn to perform the first read from that file.

As soon as the no-op at USTOPn+20 is passed, it is converted to an unconditional branch. When a particular USTOPn is entered the second time, it will be because an EOF has been detected in that input

file. This time through, the compare loop branches for the depleted file are closed off and the corresponding bit in PH2MOM is inverted to zero. As long as there are more files to merge, the branch at USTOPn+20 is to an address in the remaining compare loops which varies with current file and the order of merge; these addresses are listed on Chart EC. When all the input files are depleted and the 1-bits in PH2MOM have been inverted to 0-bits, a branch is made to the end-of-job routine (Chart EJ).

OPTIONAL ROUTINES

RELOCATOR ROUTINE - FA

Phases 2, 3, and 4 each have a relocator as part of the individual phase. The three routines are enough alike that their logic is described in this one section; any significant differences are noted in the function-block text.

The optional routines required in each phase are initialized and relocated in main storage in the minimum required area depending on the number and length of required routines. The major steps performed are:

- Initialize the relocator by turning on specified bits only for those optional routines that are required.
- Adjust the lengths of the optional routines to the next full-word boundary, if necessary.
- Relocate each required routine when necessary so that they immediately follow each other starting at the end of the mainline. In this manner, any unneeded routines, or portions of routines, are overlaid and the program then occupies the least possible amount of main storage.
- Activate branches in the selector to the proper address for a routine, if required.

The relocator then returns to the mainline initialization routine in its respective phase.

RELOCA, FA-B1

The operations performed between this location and RLFWB vary, depending on the phase in process:

- In phase 2, a branch-and-link is made to EQINIT to initialize the equal routine, if required. The move routine in the mainline is then initialized to move the specified record length (fixed-length records only). If necessary, the move routine is then relocated to the end of the mainline or, if the equal routine is included, to the end of the equal routine.

- In phase 3, in addition to initializing the move routine as described for phase 2, a bit is turned on in RLCOND for the required data reconversion routine, if any. The selected routine is then initialized by preparing the selector at STR3 with the address of the desired routine and by inserting the number of control fields. Finally, the equal routine, if required, is initialized at EQINIT.
- In phase 4, the selected data conversion/reconversion routine, if any, is initialized as described for phase 3. Then the equal routine, if required, is initialized at EQINIT. The other operations described for phases 2 and 3 are not required in the phase 4 relocator, because the move routine is assembled and executed in a different manner and the bits in RLCOND were set at CLIFOR in the initialization routine (Chart EB).

RLFWB, FA-D2

The optional routines are aligned to full-word boundaries, if necessary, and the routine continues to RLRUTR.

RLRUTR, FA-E2

Starting with the equal routine, location RLCOND is tested to determine which of the optional routines are required. If the optional routine tested for is not required, the program branches to RLMDGT to modify the relocator to test for the next optional routine.

If the routine tested for is required, its starting address is compared to the address of the first byte available for the optional routines in the main storage I/O area (RLISA).

If the two addresses are equal, no relocation is required and a branch is made to RLUPDT. The need for relocation depends on one or more of several factors:

- Whether or not a data conversion routine is required, and if it is, which one and for how many control fields.

- Whether or not the equal routine is required.
- The length of the move routine (phases 2 and 3 only).

If relocation is required, the routine is moved from its old address (original) to its new address (starting at the address contained in RLISA), 256 bytes at a time. After the last move (which may contain less than 256 bytes) is executed, the selector branch is updated for the routine just moved. The total length count (counting from the end of the mainline program) is increased by the length of the routine just included. The relocator then continues to RLUPDT.

RLUPDT, FA-C3

The starting address of the I/O area (RLISA) is increased by an amount equal to the length of the optional routine just included in the program. This will be the new starting address for the next optional routine that may be required or, if no more are required, this will be the starting address of the I/O areas.

RLMDGT, FA-H3

The relocator is modified to process the next optional routine. This includes:

- The location of the length of the next routine.
- The testing mask at RLINVR for determining if the next sequential routine is required.
- The constant used for updating the testing mask.

RLSTRP, FA-J3

The relocator branches back to RLRUTR to process the next optional routine until all the assembled optional routines have been checked and processed. Control is then returned to the mainline initialization routine:

- In phase 2, at START in Chart CA.
- In phase 3, at START in Chart DA.
- In phase 4, at INIT3 in Chart EB.

EQINIT, FA-B4

In phases 3 and 4, a test is made to determine the number of control fields. For more than one control field, register 9 is initialized with the address of the control field table (CF1LCT-1) and the sequence of the first control field is saved. If the record contains only one control field, the program branches to ANONE to initialize the data conversion routines for one control field.

Note: In phase 2, ANONE is not used (no data conversion in phase 2).

In phases 2, 3, and 4, register 8 is then loaded with the number of control fields, and the routine continues to ANIVTA.

ANIVTA, FA-D4

The displacement and length of the control field are inserted in the compare set in the equal routine. The first time through, both operands are initialized for the first control field only. On subsequent passes (one for each control field), the operands for the other control fields are initialized.

A test is then made to determine if the sequence of the current control field is the same as that of the first control field. If it is, the equal routine base registers are placed in ascending order in the first and second operands of the compare instruction for that control field. If the sequence is opposite to that of the first control field, the base registers are reversed in the operands. The equal routine is thus initialized to sort each control field in the specified sequence.

ANIULT, FA-G4

The addresses of the next control field location and of the next compare instruction in the equal routine compare set are obtained. The routine then branches back to ANIVTA to initialize the compare for the next control field. When processing for all control fields is complete, the length of the equal routine is computed and inserted in RLNG2. Control is then returned to the relocator at label RLFWB.

FIXED-POINT CONVERT/RECONVERT ROUTINE - FB

This routine converts the control data fields from fixed-point format to binary by inverting the sign of each control field.

FIXPNT, FB-B3

A register is initialized with the starting address of the compare set. (Compare set at IGLCOM in the equal routine contains the length and the displacement of the control fields.)

FIXR2, FB-C3

The number of control fields (established by the fixed-point initialization routine) is loaded into a register. This number is used to end the conversion routine after all control fields have been processed.

FIXVTA, FB-D3

The address of the control field to be converted is calculated and the sign bit is inverted. The address is modified to determine the starting address of the next control field, and the count of the number of control fields is reduced by one. The program returns to the instruction at the label FIXVTA, and the process is repeated until the count of the number of control fields is reduced to zero and control is returned to the mainline.

Reconversion follows the same procedure, thus restoring the inverted sign bit of each control field to its original form.

FLOATING-POINT CONVERT/RECONVERT ROUTINE - FC

This routine converts the control data fields from floating-point format to binary. The program modifies negative and

positive floating-point fields within each record so that they can be used in the compare operations of the sort program.

FLTPNT, FC-B1 (FOR CONVERSION ONLY)

Registers are initialized with the address of the compare set and the number of control fields to be converted.

FLTVTA, FC-F1

The address of the first control field is determined. If the sign of this field is positive, the sign only is inverted (FLTCSG). If the sign is negative, the entire control field, including the sign, is complemented (FLTCML).

FLTBCK, FC-E3

The starting address for the next control field is determined, and the count of the number of control fields is reduced by one. The program returns to the instruction at the label FLTVTA, and the process is repeated until the control field count becomes zero, indicating that all the control fields have been converted. Control is returned to the mainline via the link register in phases 2 and 4; phase 3 branches to EXIT32 in the output routine.

To reconvert the control fields to the floating-point format, the reconversion switch (FLTSWH) is set, reversing the procedure in FLTVTA and restoring the control field to its original form (Figure 51).

	Control Field Sign	Reconversion Switch	Invert	Converted Field Sign
Conversion	+ -	OFF OFF	Sign bit Field	- +
Reconversion	- +	ON ON	Sign bit Field	+ Restored - Sign

Figure 51. Floating-Point Conversion

PACKED-DECIMAL (SIGPAK) CONVERT/RECONVERT ROUTINE - FD

This routine converts control data fields from packed-decimal format to binary. The sign is moved to the high-order half byte of the control field, and the entire field is shifted (one byte at a time) a half byte to the right. The sign is always reversed and, if the number is negative, the entire control field is complemented.

SIGPK2, FD-C1

The address of the compare set is loaded into a register.

SIGKR2, FD-D1

The number of control fields (obtained during initialization of packed-decimal conversion routine) is loaded into a register. This number is used to end the conversion routine after all control fields have been processed.

SIGCP4, FD-E1

The address of the sign byte is calculated, and the sign bits are stored in PKSGN2. The sign bits are tested for a negative or a positive value (in either EBCDIC or ASCII code). If positive, a true positive sign (hex C for EBCDIC or hex A for ASCII) is generated.

PAKLP2, FD-E2

The digit portion (two hex digits) is shifted a half byte to the right and is complemented if the sign value (PKSGN2) is negative.

A test is made to determine if the left-most byte of the control field (high-order position) has been reached. If not, the byte address is modified to obtain the next byte to the left, and the program returns to PAKLP2 to repeat the process for the next byte. If the high-order byte is detected (indicating the end of the control field), the sign bits are retrieved from PKSGN2 and placed in the leftmost 4 bits of the converted control field. The sign is then reversed.

The control-field address is modified for the next field to be converted. Each control field is converted in the same manner and, when the last field has been converted, control is returned to the mainline via the link register.

Reconversion takes place in a similar manner, except that the digit portion is shifted left (instead of right) a half byte, and the sign bits are restored to the low-order byte of the control field.

ZONED-DECIMAL (SIGZON) CONVERT/RECONVERT ROUTINE - FE

This routine converts control data fields from zoned-decimal format to binary. The sign of the control field is saved, and the leftmost zone is moved to the sign position. The sign is then inserted in the leftmost zone of the control field and reversed. If the number is negative, the entire field is complemented.

SIGZN2, FE-B1

Registers are initialized with the address of the compare set and the number of control fields to be converted.

SIGCP2, FE-D1

The address of the last, low-order, byte of the control field is calculated and the sign is stored in ZONZN2. The leftmost (high-order) zone bits are moved to the position previously occupied by the sign bits, and the content of ZONZN2 is tested for a positive or negative value (in either EBCDIC or ASCII code). If positive, a true positive sign (hex C for EBCDIC or hex A for ASCII) is generated in ZONZN2. In either case, the sign is then moved to the position vacated by the zone bits. Once moved, the sign is reversed, and a branch is made to ZNLST2.

ZONPS2, FE-E4

If ZONZN2 is negative, the digit bits are complemented, and the byte address is updated for the next byte to the left. The program returns to ZONPS2 to invert the digit portion of each byte (one byte at a time) until the high-order byte is processed. The routine then continues to ZNLST2.

ZNLST2, FE-H4

When the last byte of the control field has been processed, the control field address is modified for the next field, and the routine returns to SIGCP2 to convert the remaining control fields in the same manner.

When all control fields have been processed, control is returned to the mainline via the link register.

The control fields are reconverted from binary to zoned-decimal format by:

- Restoring the sign to the original position and reversing it to its original state.
- Restoring zone bits to their original locations.
- Inverting the digit portions of each byte, if necessary.

EQUAL ROUTINE - FF

This routine compares record control fields subsequent to the first if the first fields are found to be equal when compared in the mainline.

The compare set was previously initialized for proper sequencing of each control field. As a result, if any fields are now found to be unequal, the proper sequence will be observed upon returning to the mainline conditional branches. If all fields are equal, the return point in the mainline is:

- In phases 2 and 3, to the branch instruction preceding the BAL or BALR to this routine.
- In phase 4, to the next sequential instruction after the BAL to this routine.

ROUT1, FF-B3

Note: In phases 2 and 3, this label is EQUAL.

Entry is made to this point when the first control fields of two records were found to be equal in the mainline compare loops. The link register (register 14) is adjusted to the address of the mainline compare instruction from which this routine was entered.

The base registers from the mainline compare instruction are determined and inserted in the instructions at IGCAR1 and IGCAR2. These two instructions then load the registers in the operands of the compare set of this routine. Note that the compare set was initialized with its base registers in the proper operand in each compare, depending on whether the sequence for that control field was the same as or different from that of the first control field. In this way, each of the required control fields will be compared for the proper sequence, regardless of whether it is ascending or descending.

IGLCOM, FF-D3

The corresponding control fields in the two records are compared until either an unequal condition is found or the end of the compare set is reached. In the former case, a branch is made (via IGOUT) back to the conditional branches following the

mainline compare instruction to continue processing the record. In the latter case, when all corresponding fields within the two records were found to be equal, the address for returning to the mainline is adjusted (at IGDUM):

- In phases 2 and 3, to the branch instruction preceding the BAL or the BALR to this routine.

- In phase 4, to the next sequential instruction after the BAL to this routine.

A branch is then made (via IGDUM) back to the mainline to continue processing.

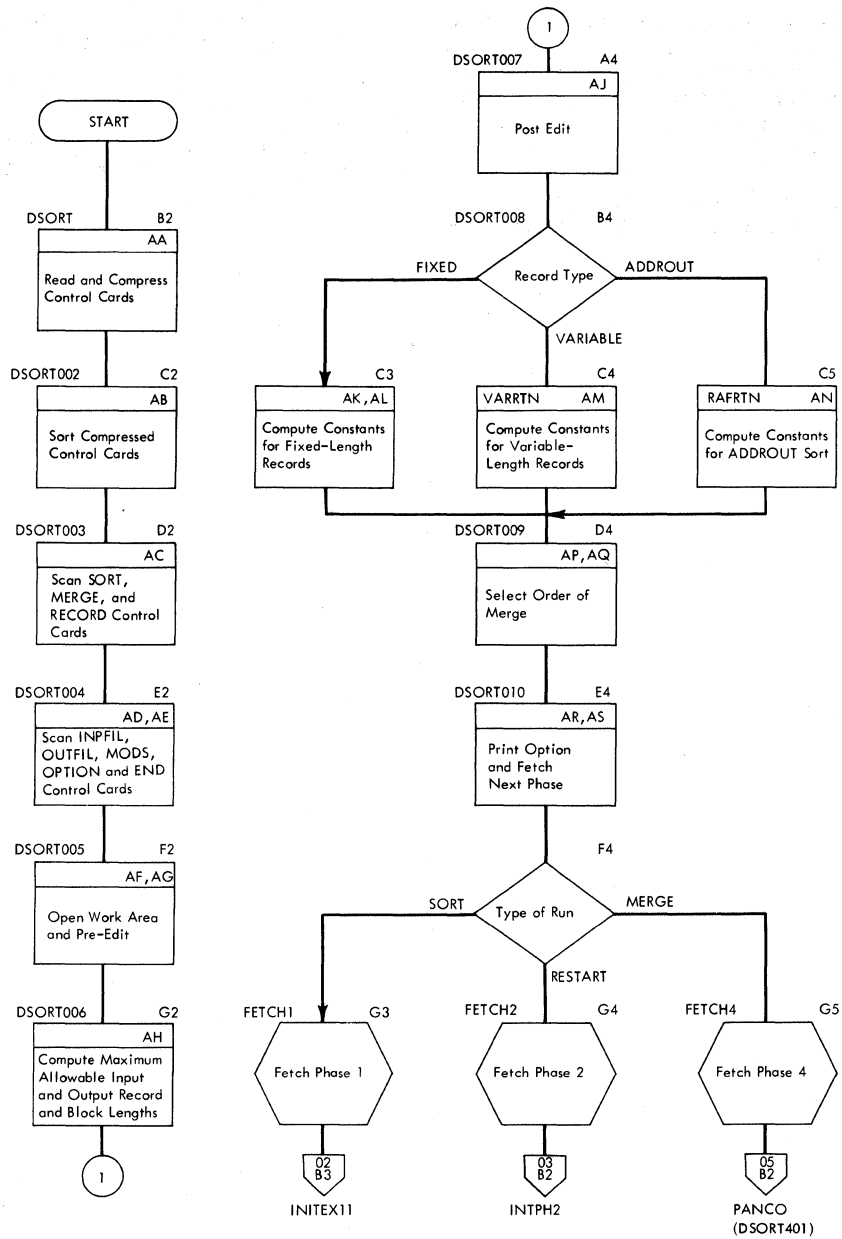


Chart 01. Assignment Phase (Phase 0)

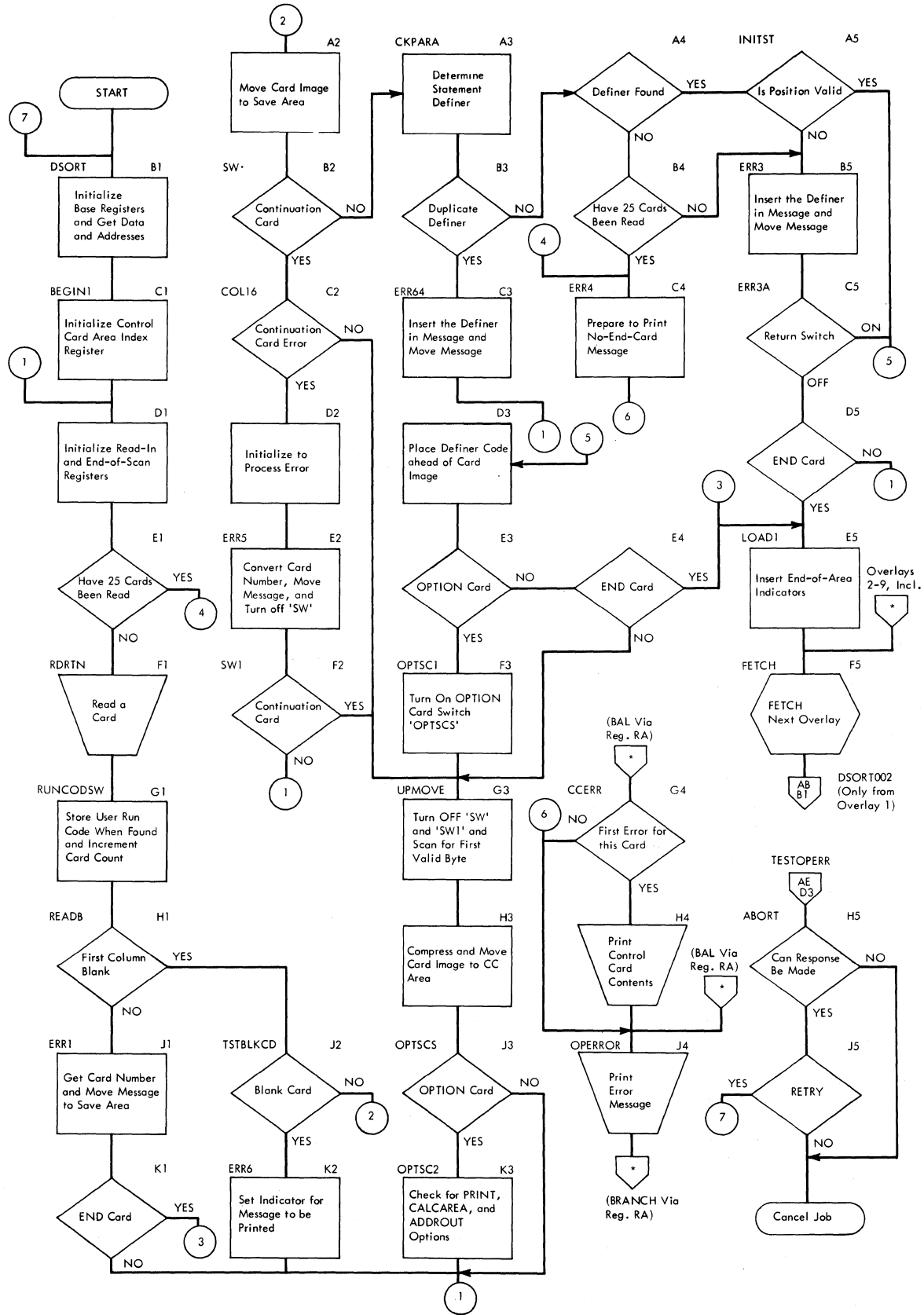


Chart AA. Read and Compress Control Cards, DSORT

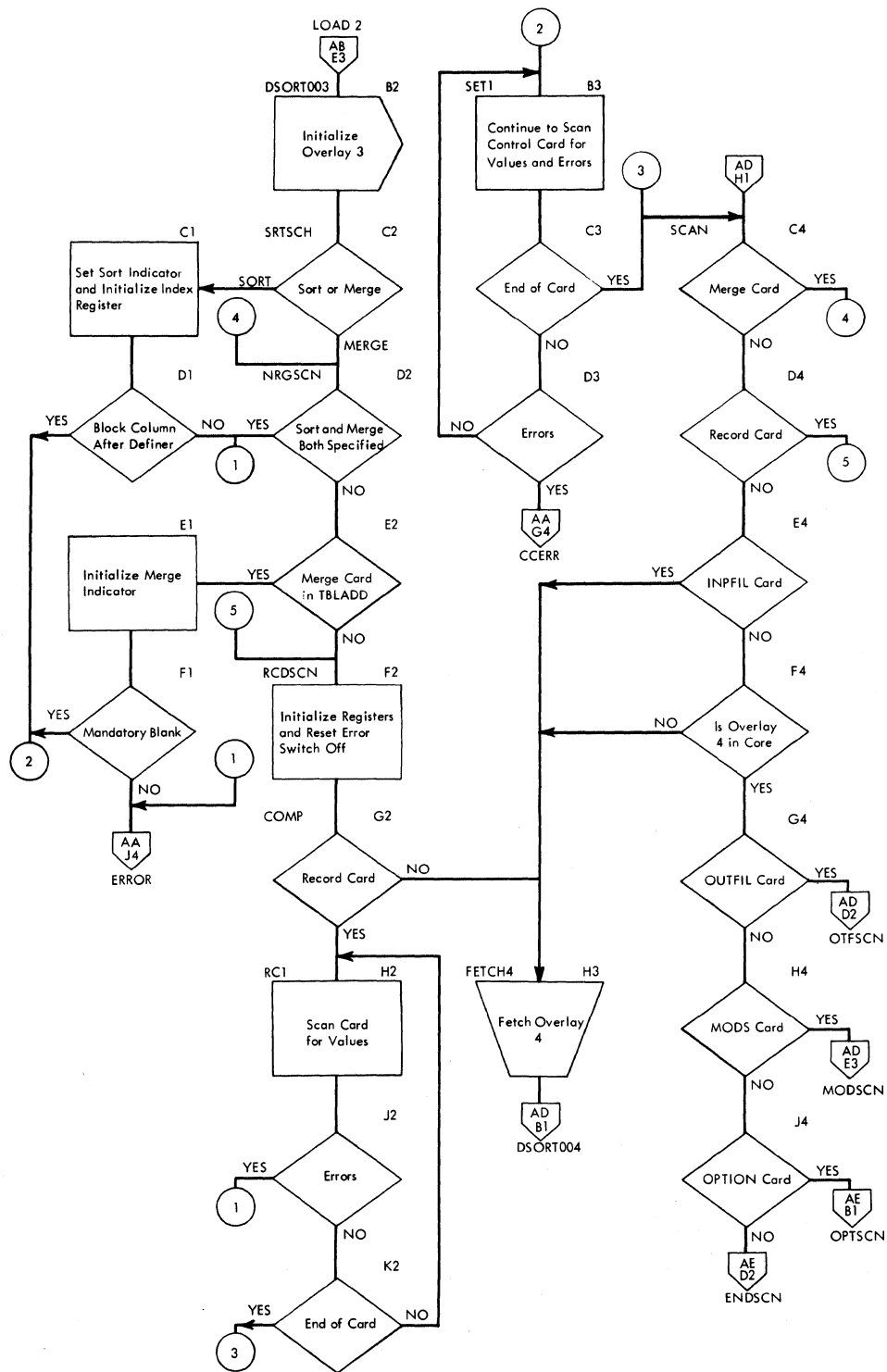


Chart AC. Scan SORT, MERGE, and RECORD Control Cards, DSORT003

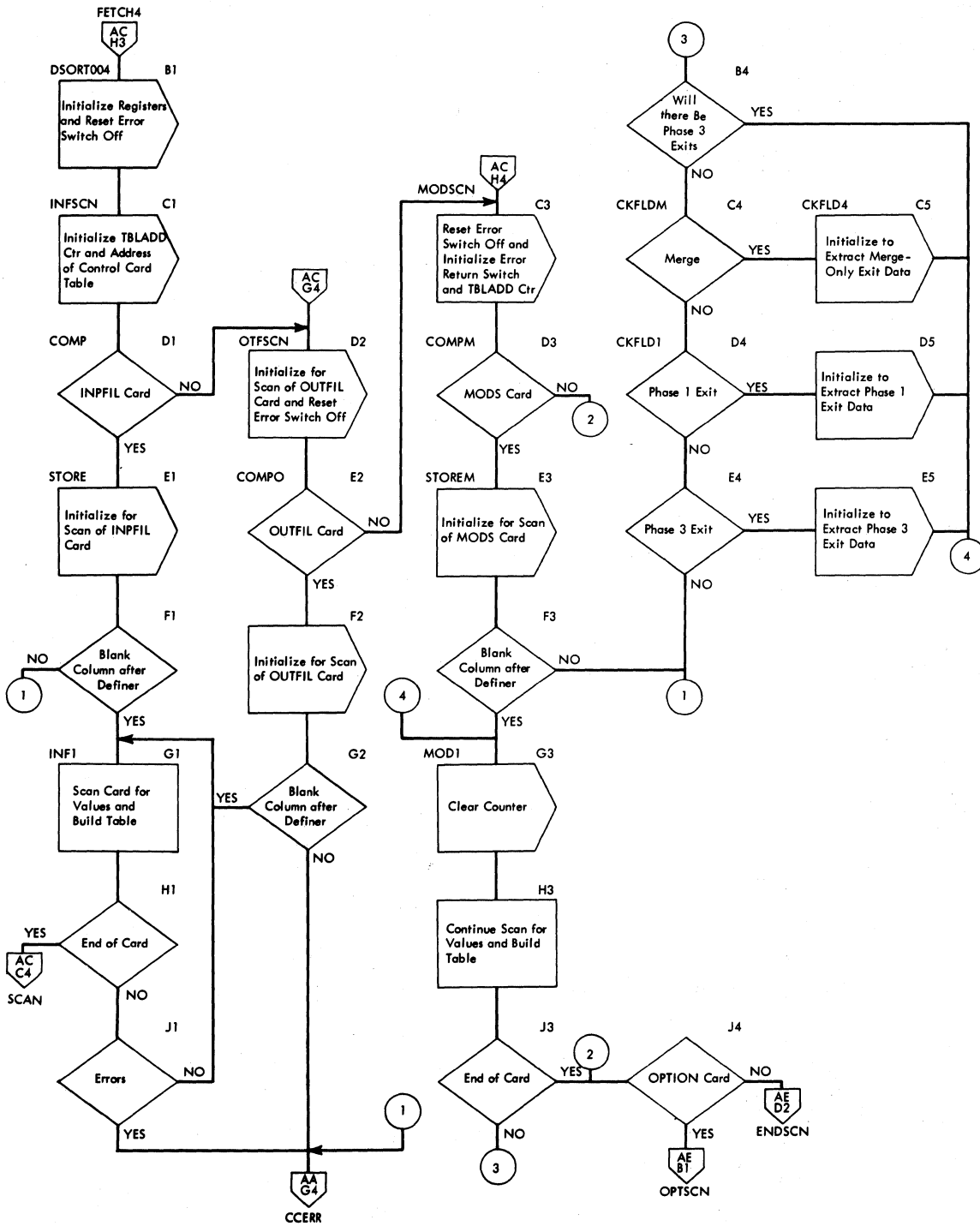


Chart AD. Scan INPFIL, OUTFIL, MODS, OPTION and END Control Cards, DSORT004

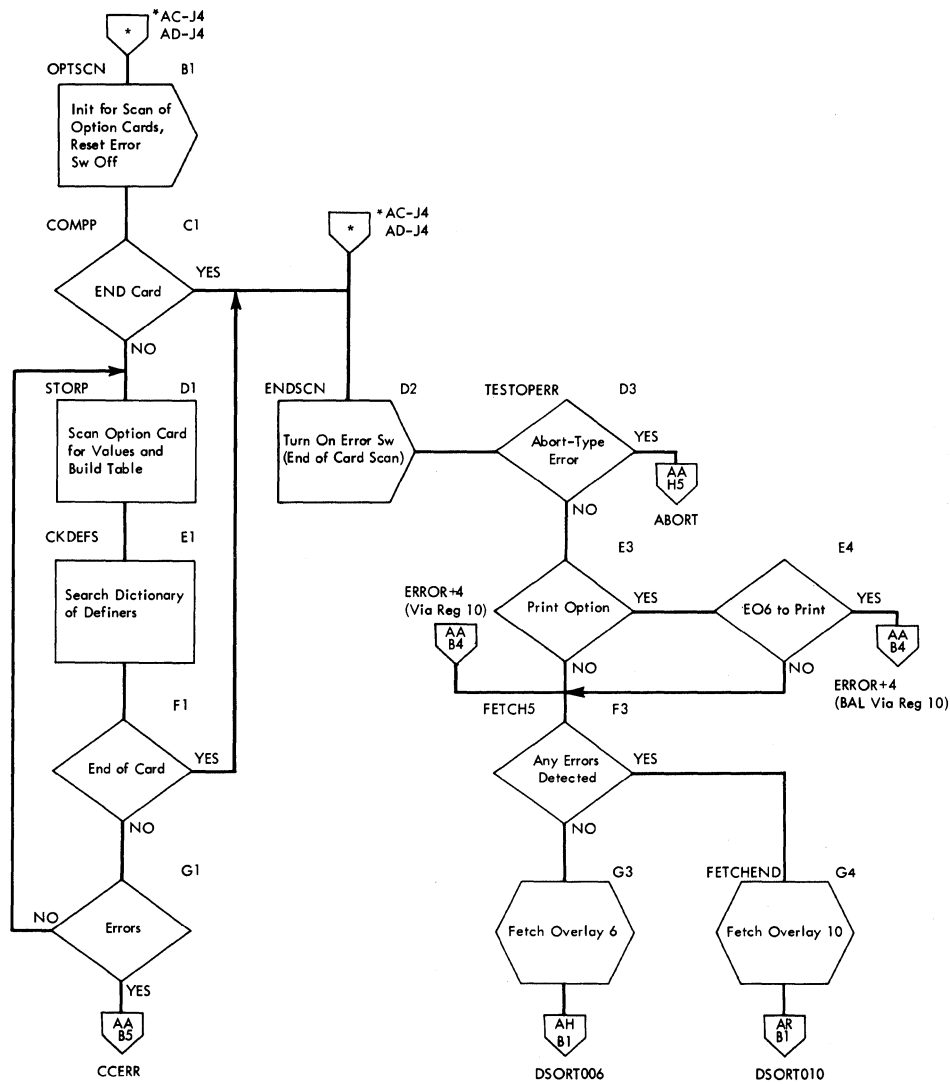


Chart AE. Scan INPFIL, UTFIL, MODS, OPTION and END Control Cards (Cont'd), DSORT004

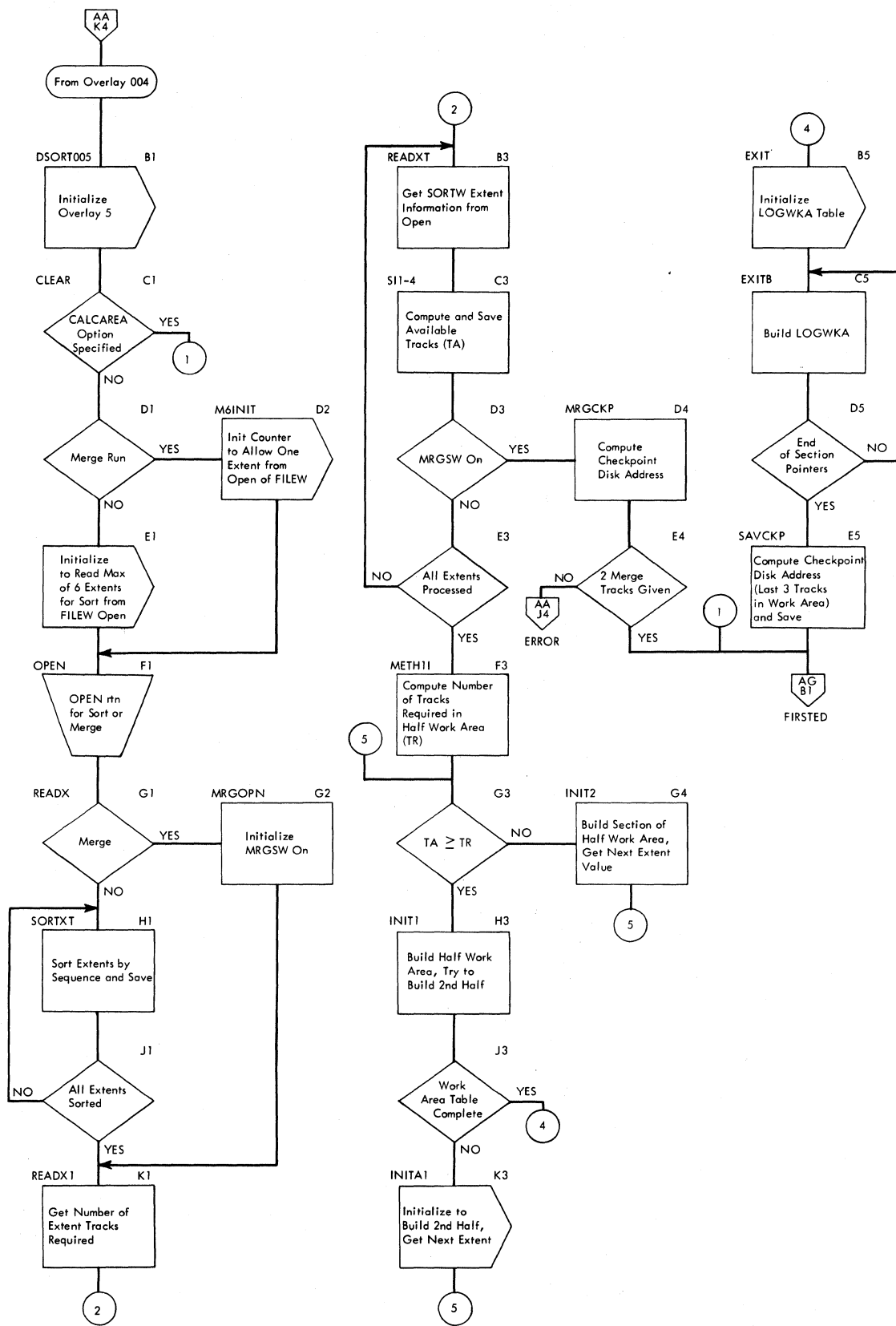


Chart AF. Open Work Area and Pre-edit, DSORT005

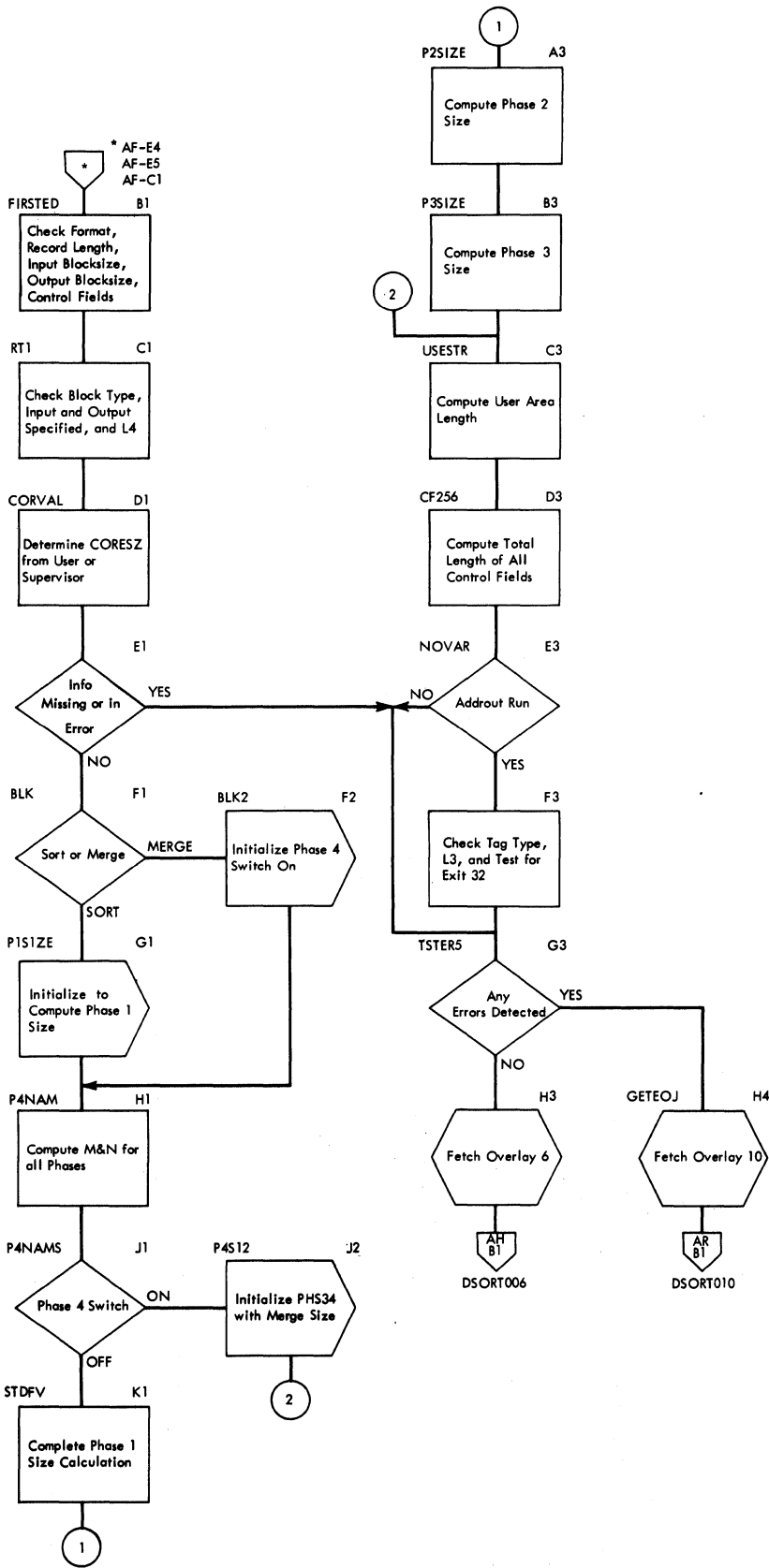


Chart AG. Open Work Area and Pre-edit (Cont'd), DSORT005

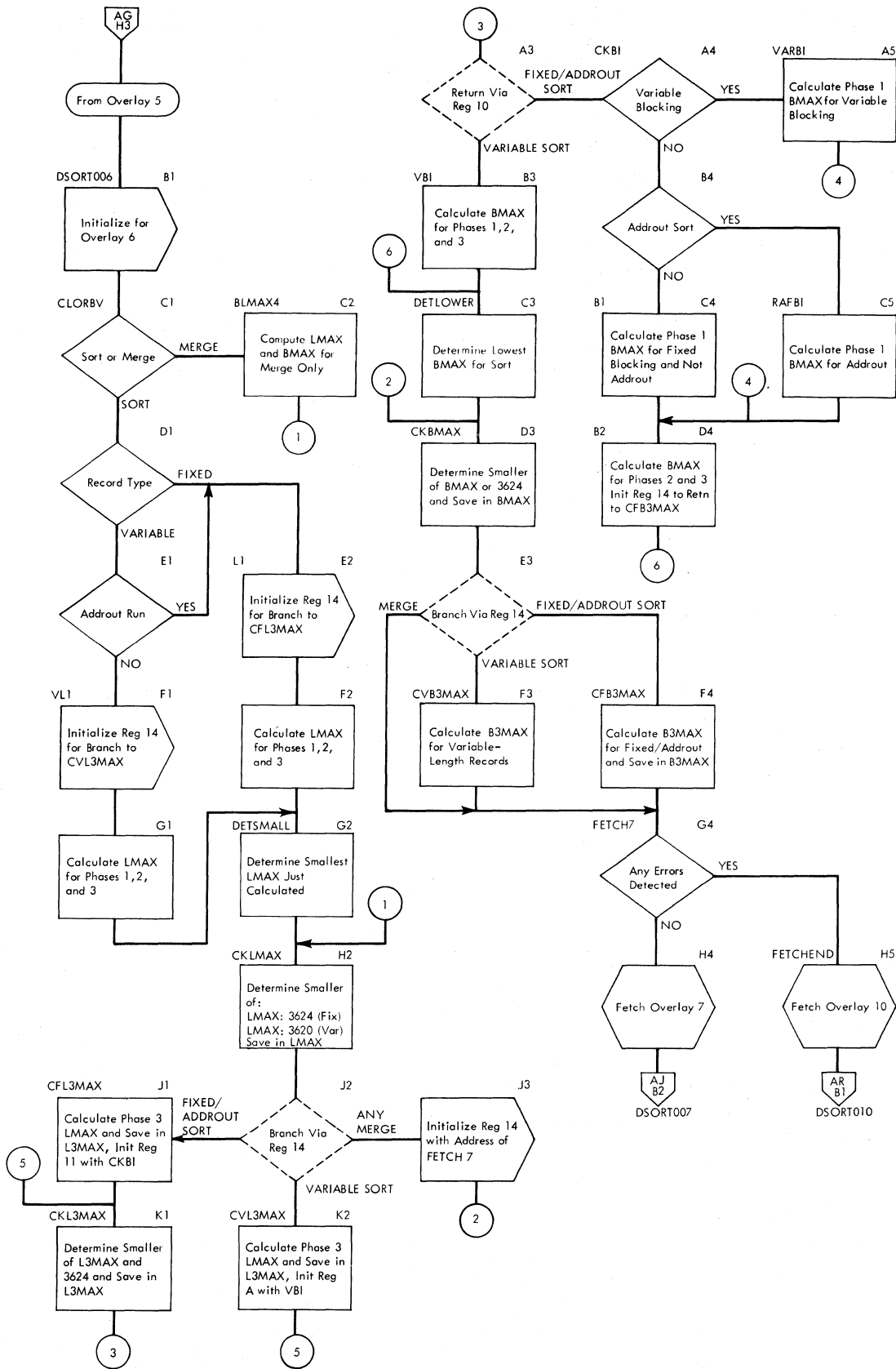


Chart AH. Compute Maximum Allowable Input and Output Record and Block Lengths, DSORT006

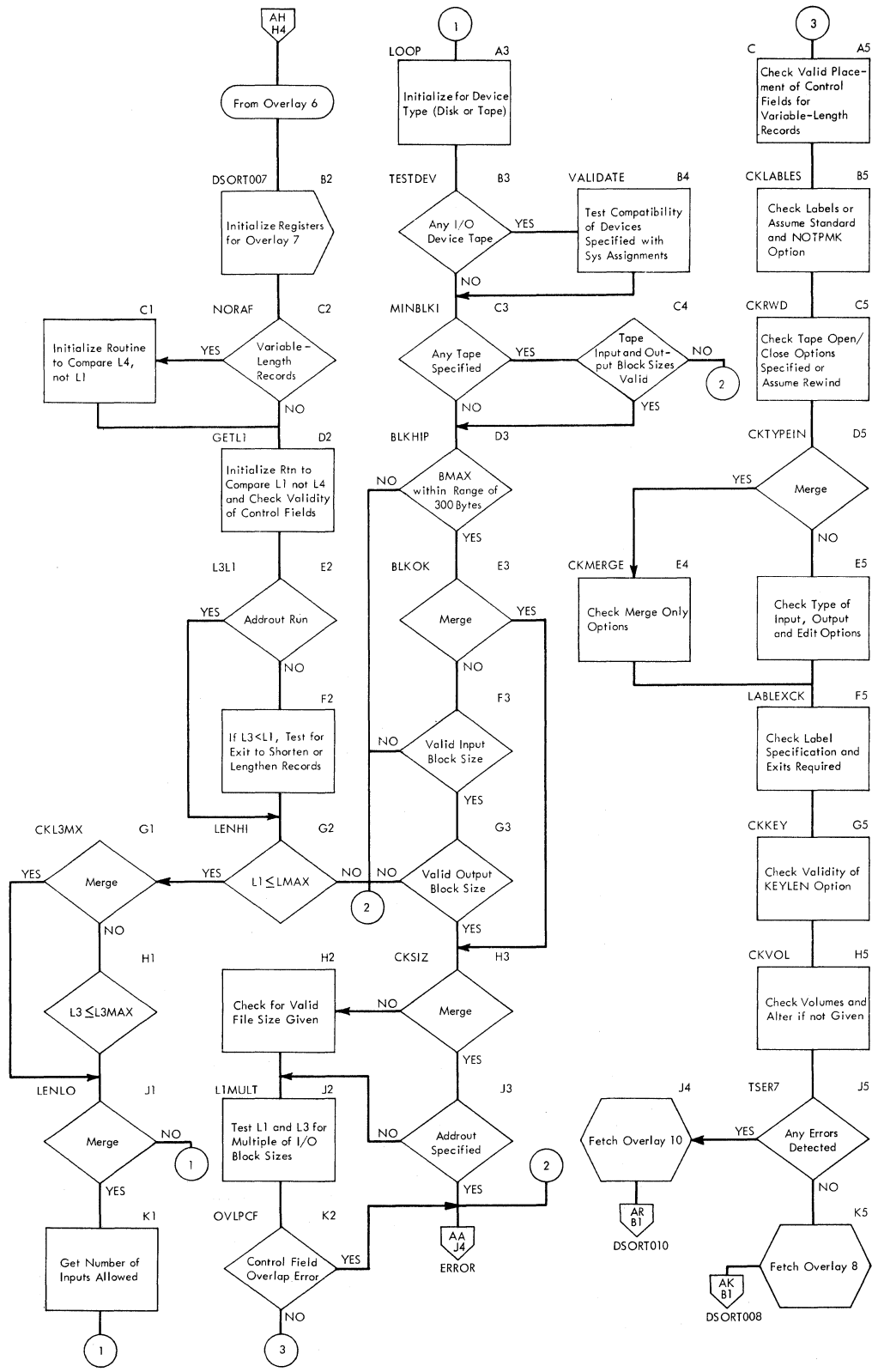


Chart AJ. Post Edit, DSORT007

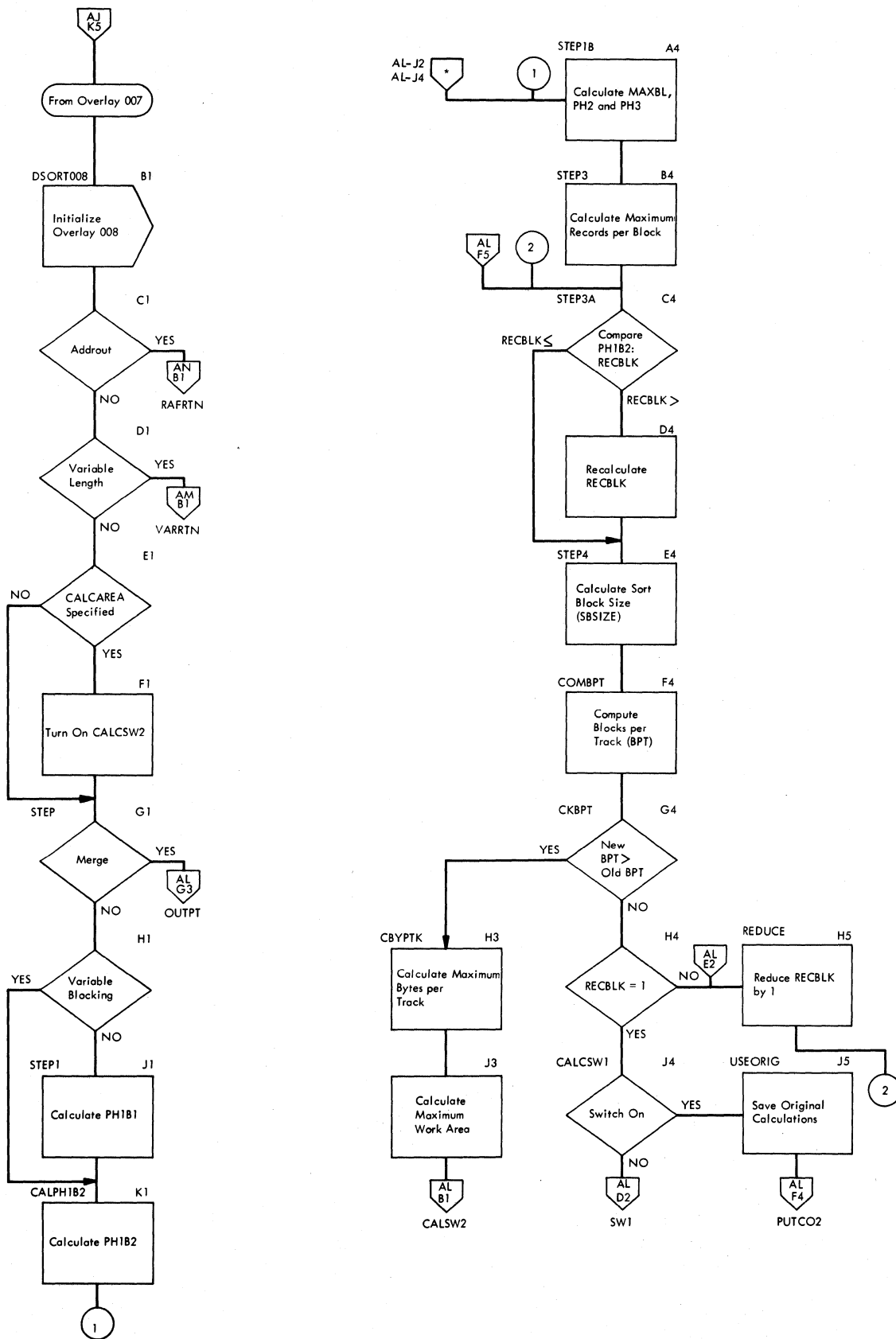


Chart AK. Compute Constants, DSORT008

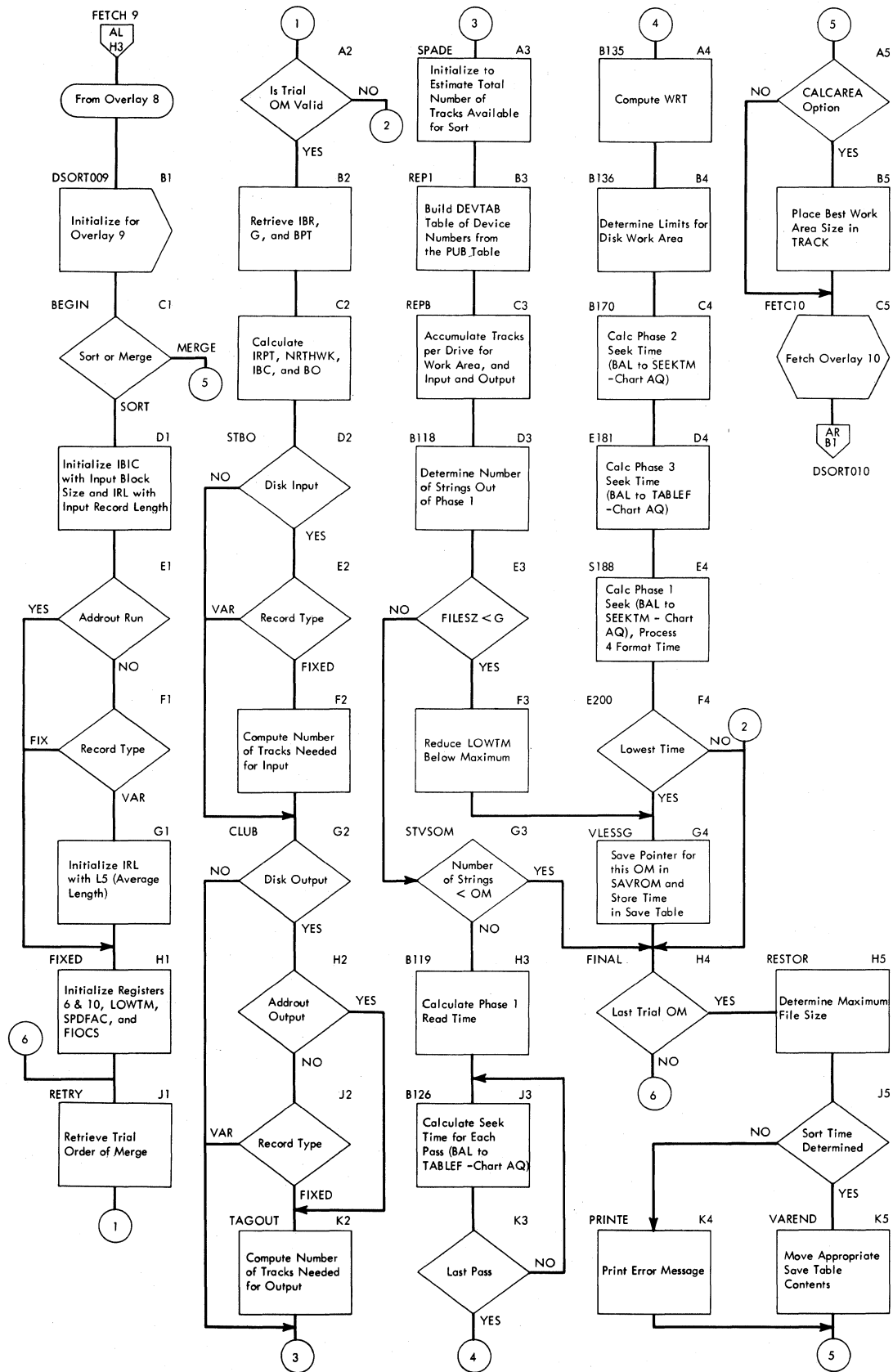


Chart AP. Select Order of Merge, DSORT009

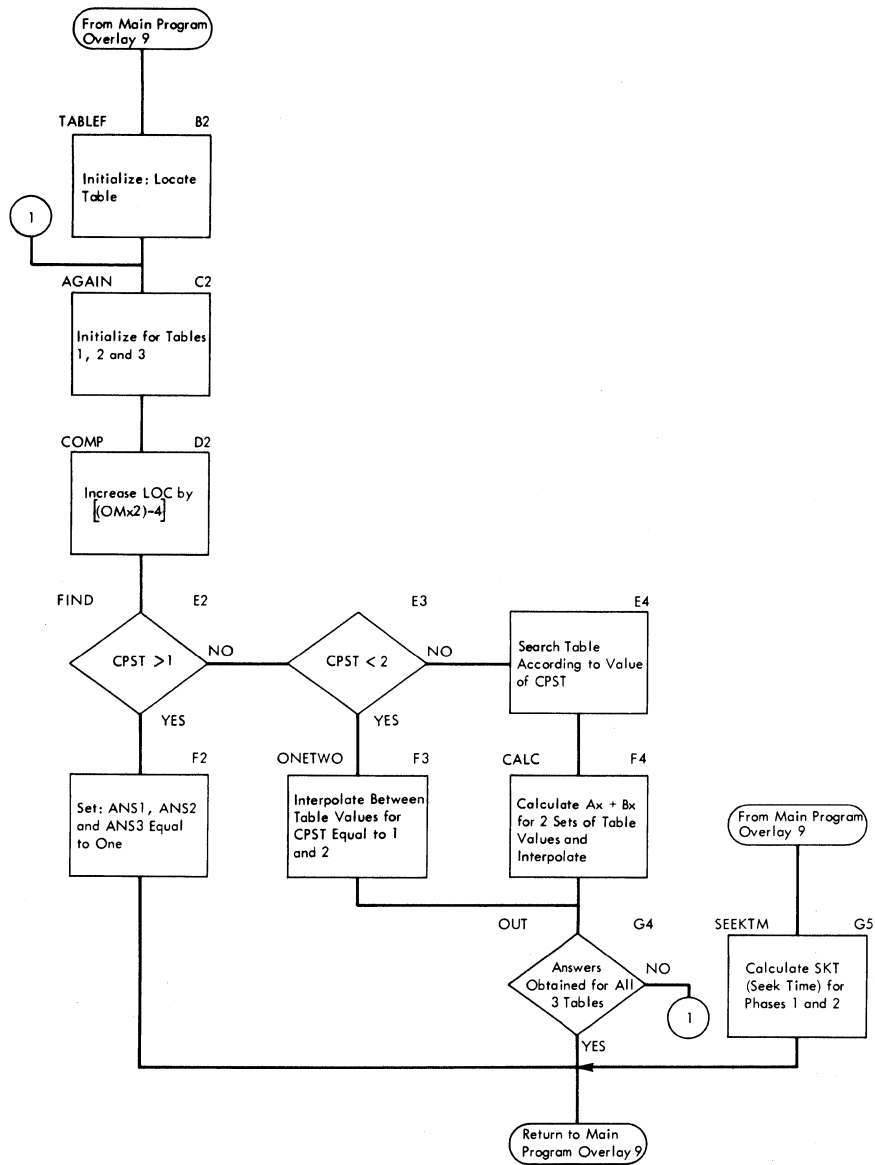


Chart A9. Select Order of Merge (Cont'd), DSORT009

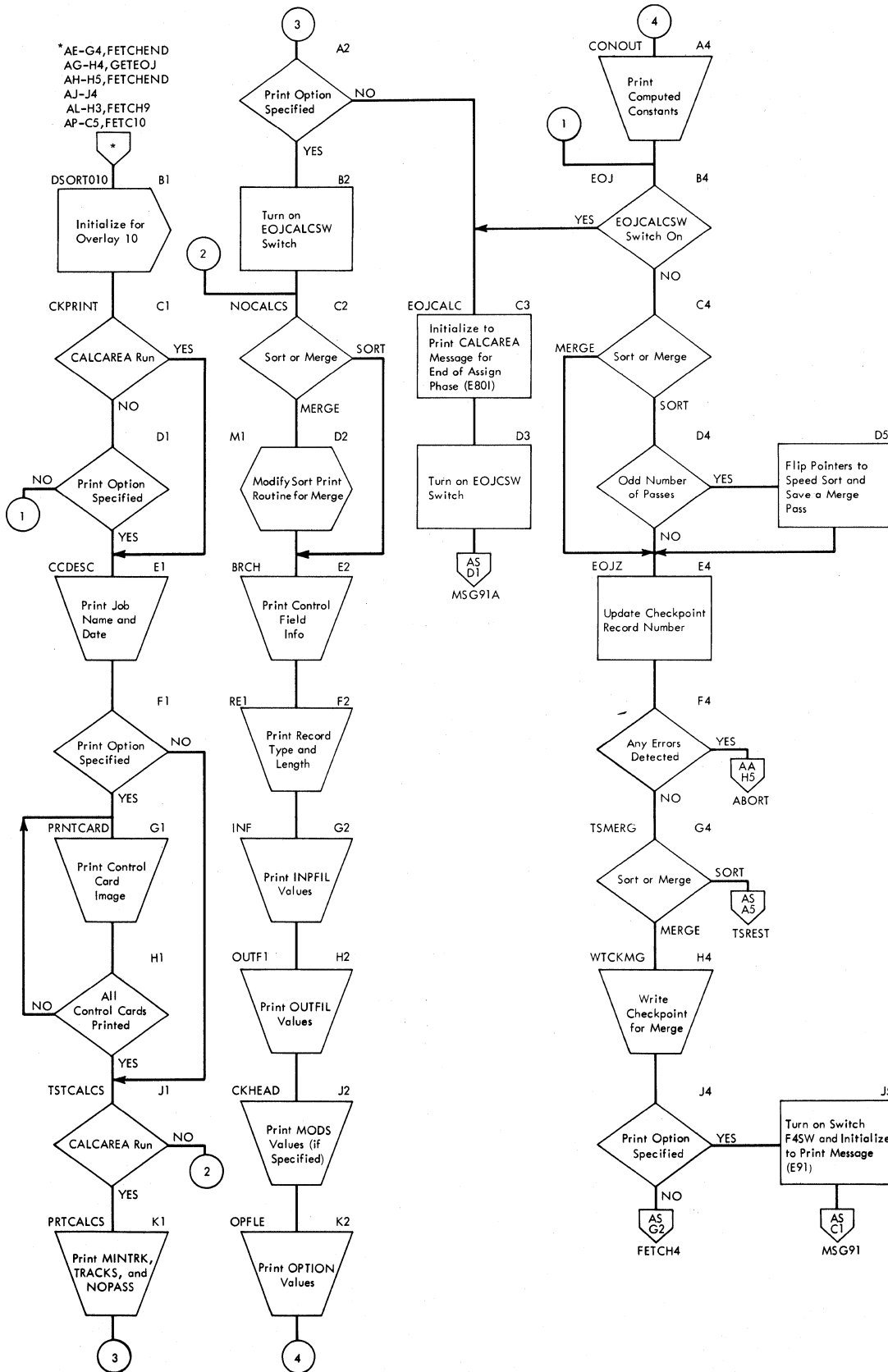


Chart AR. Print Option and Fetch Next Phase, DSORT010

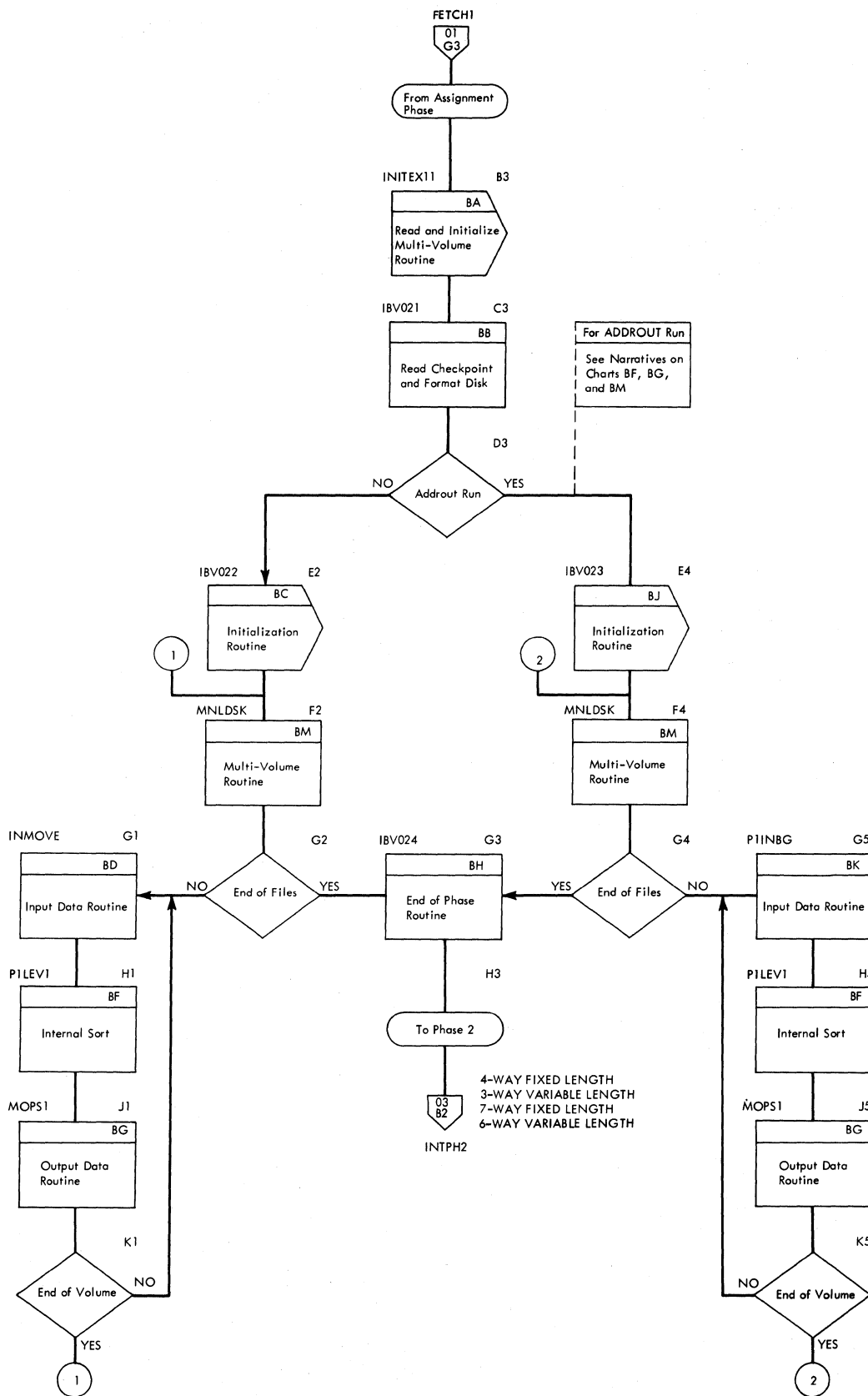


Chart 02. Internal Sort (Phase 1)

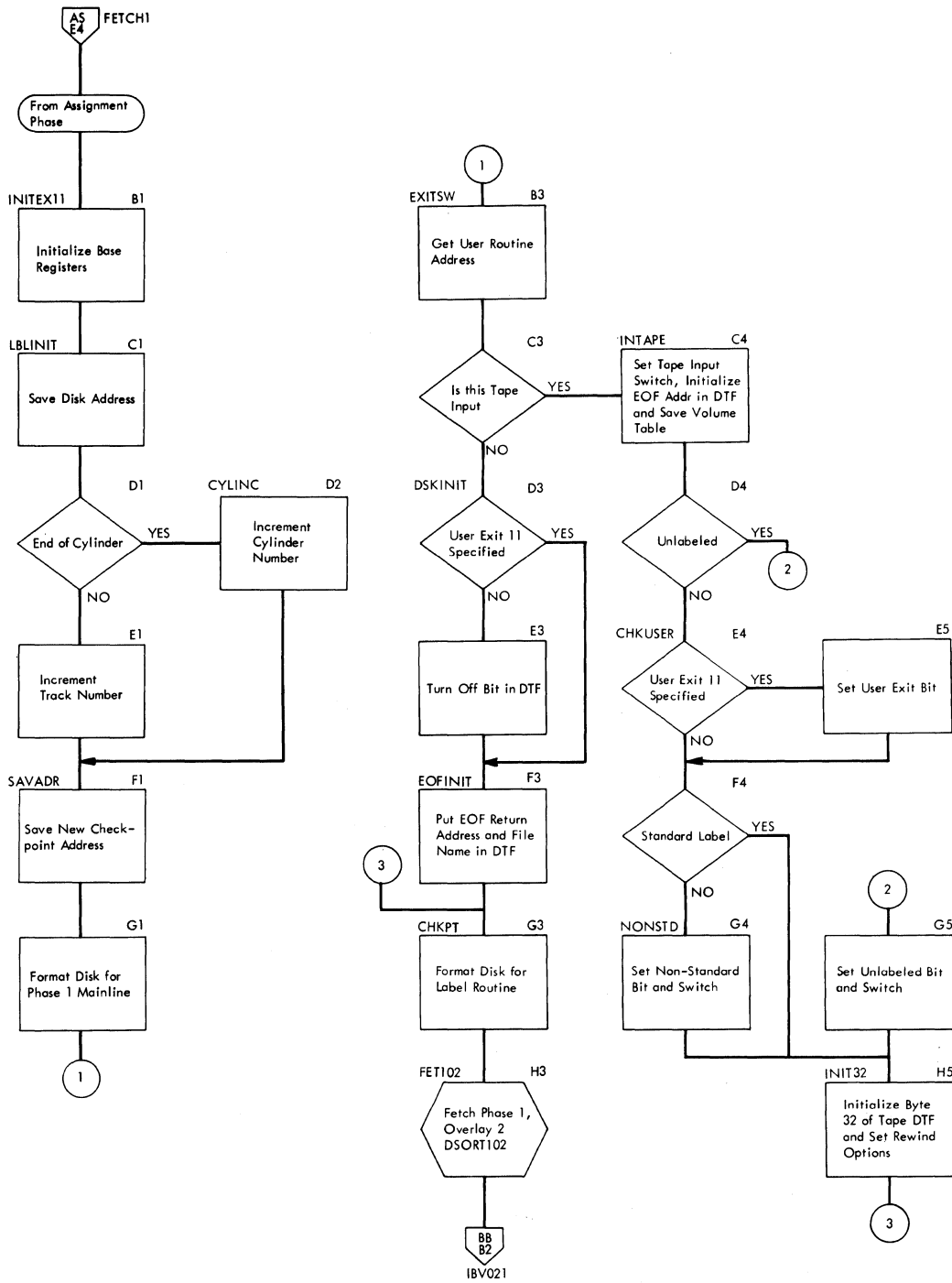


Chart BA. Initialization for Multi-Volume (Exit 11 Linkage), DSORT101

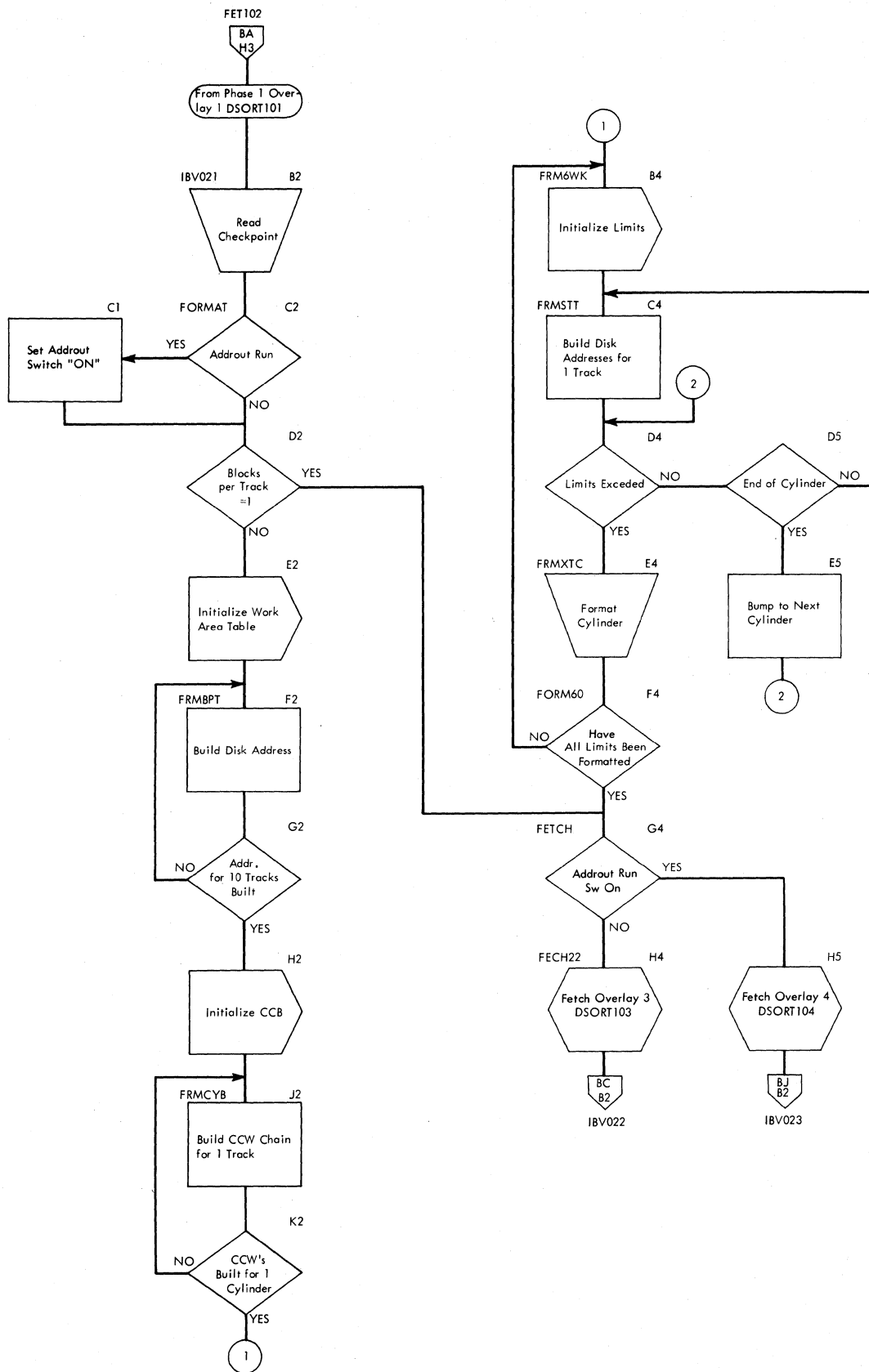


Chart BB. Format Routine, DSORT102

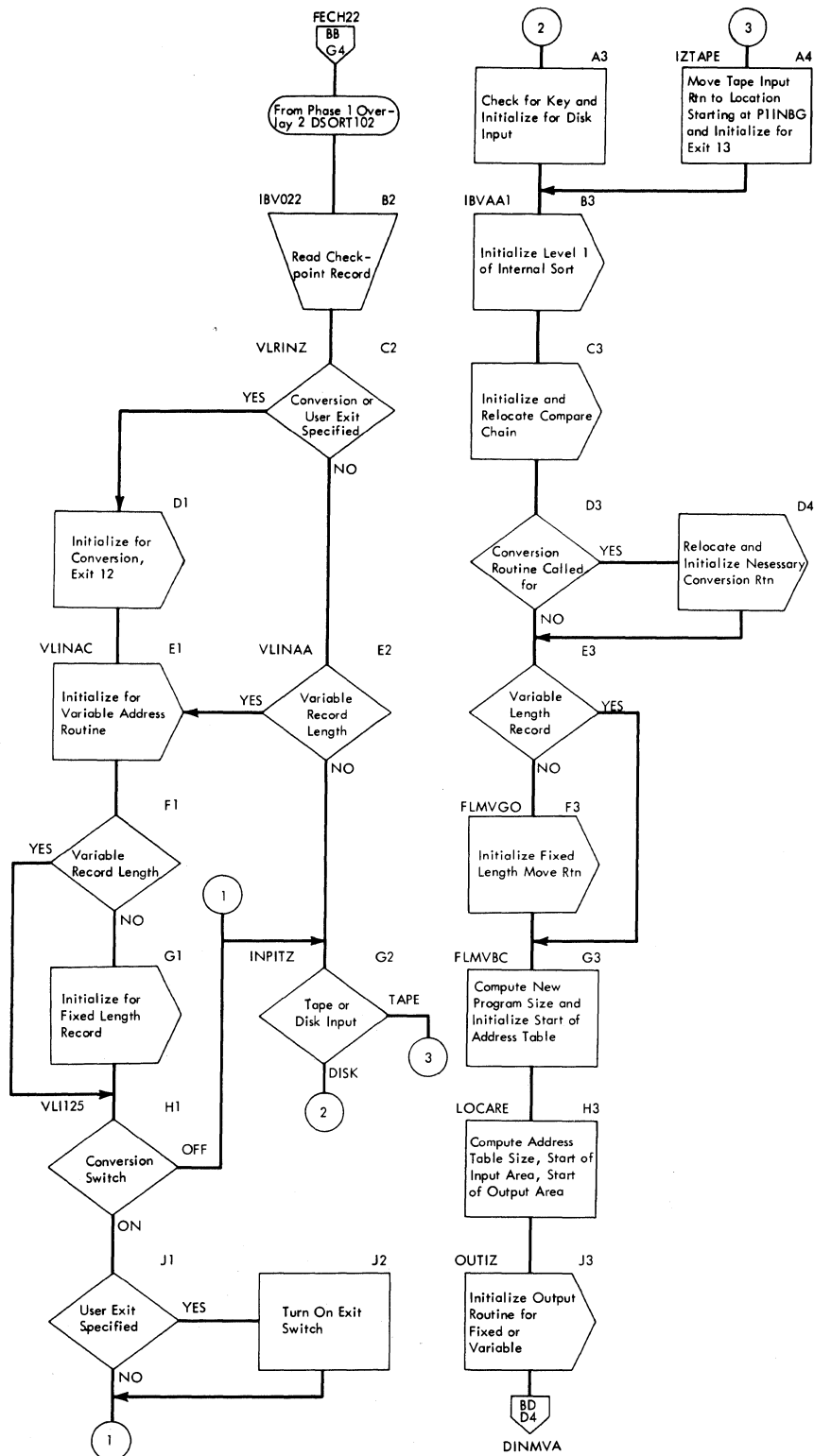


Chart BC. Initialization Routine for Disk or Tape Input, DSORT103

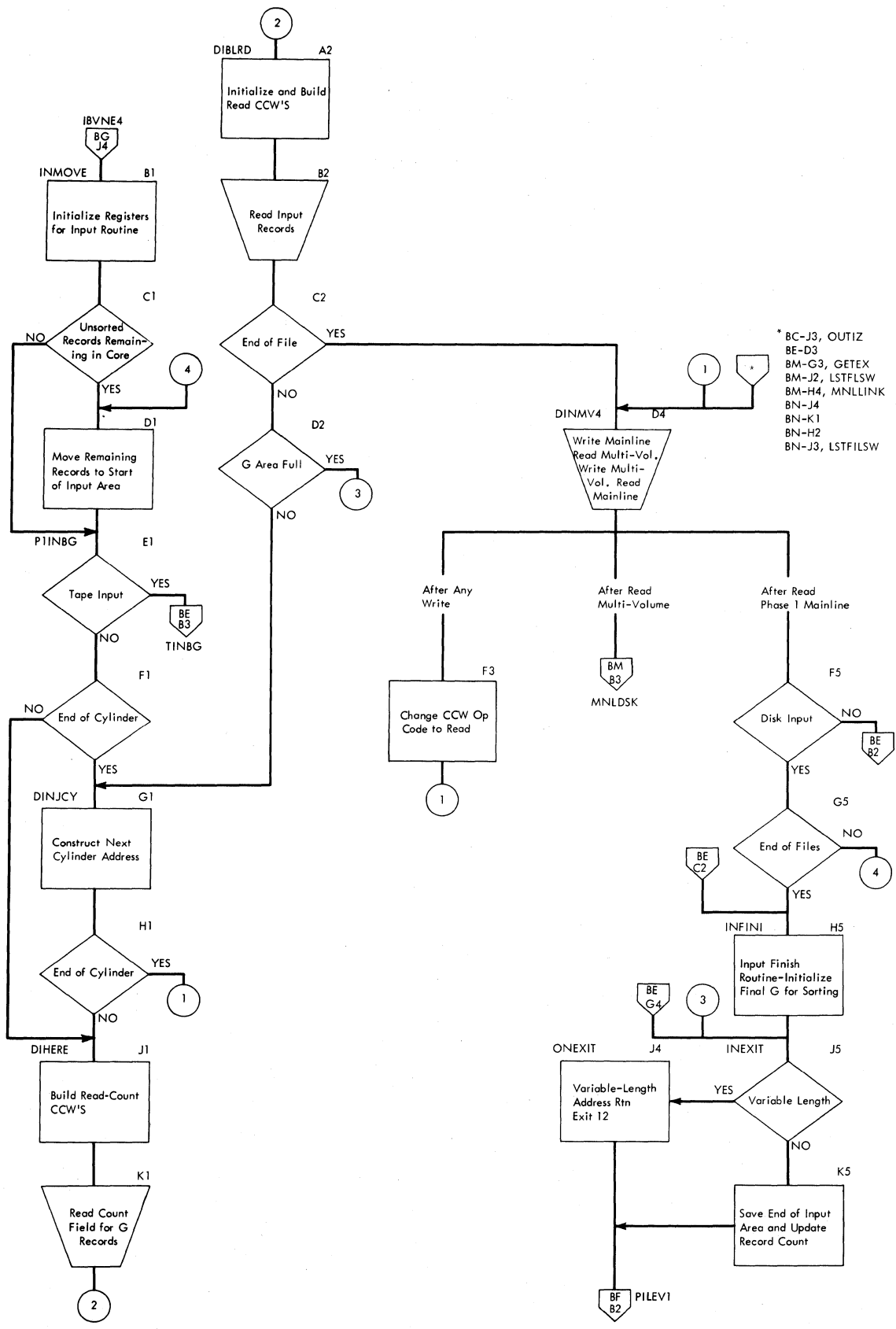


Chart BD. Input Routine for Disk or Tape Input, DSORT103

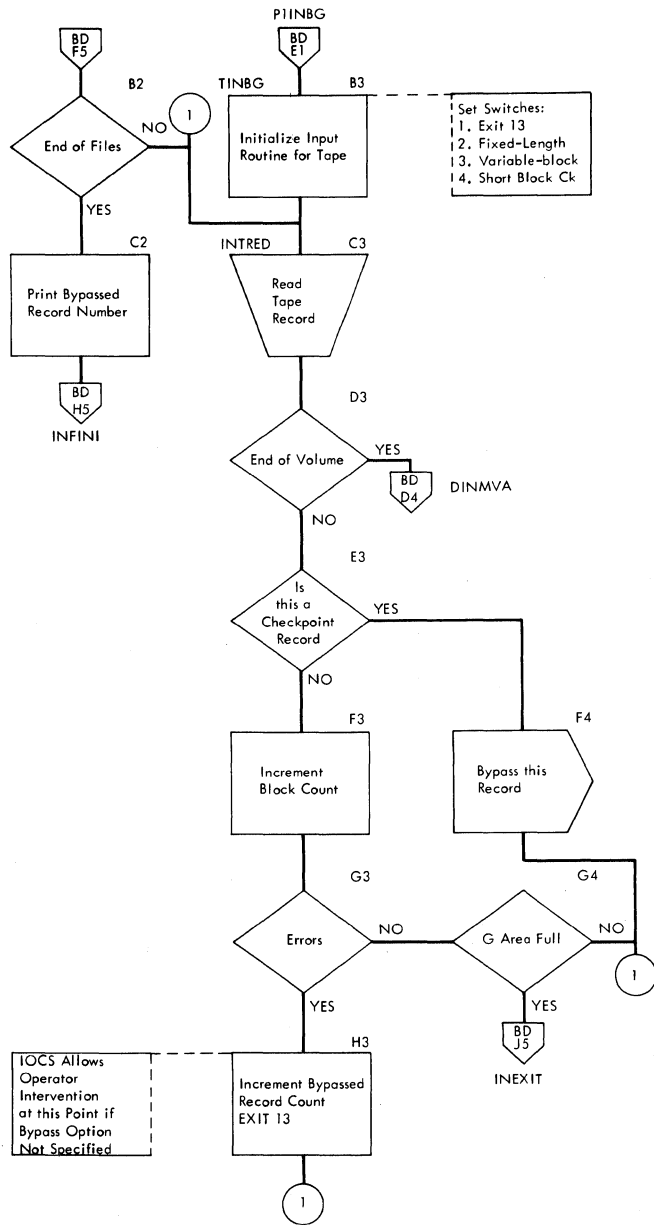


Chart BE. Tape Input Routine, DSORT103

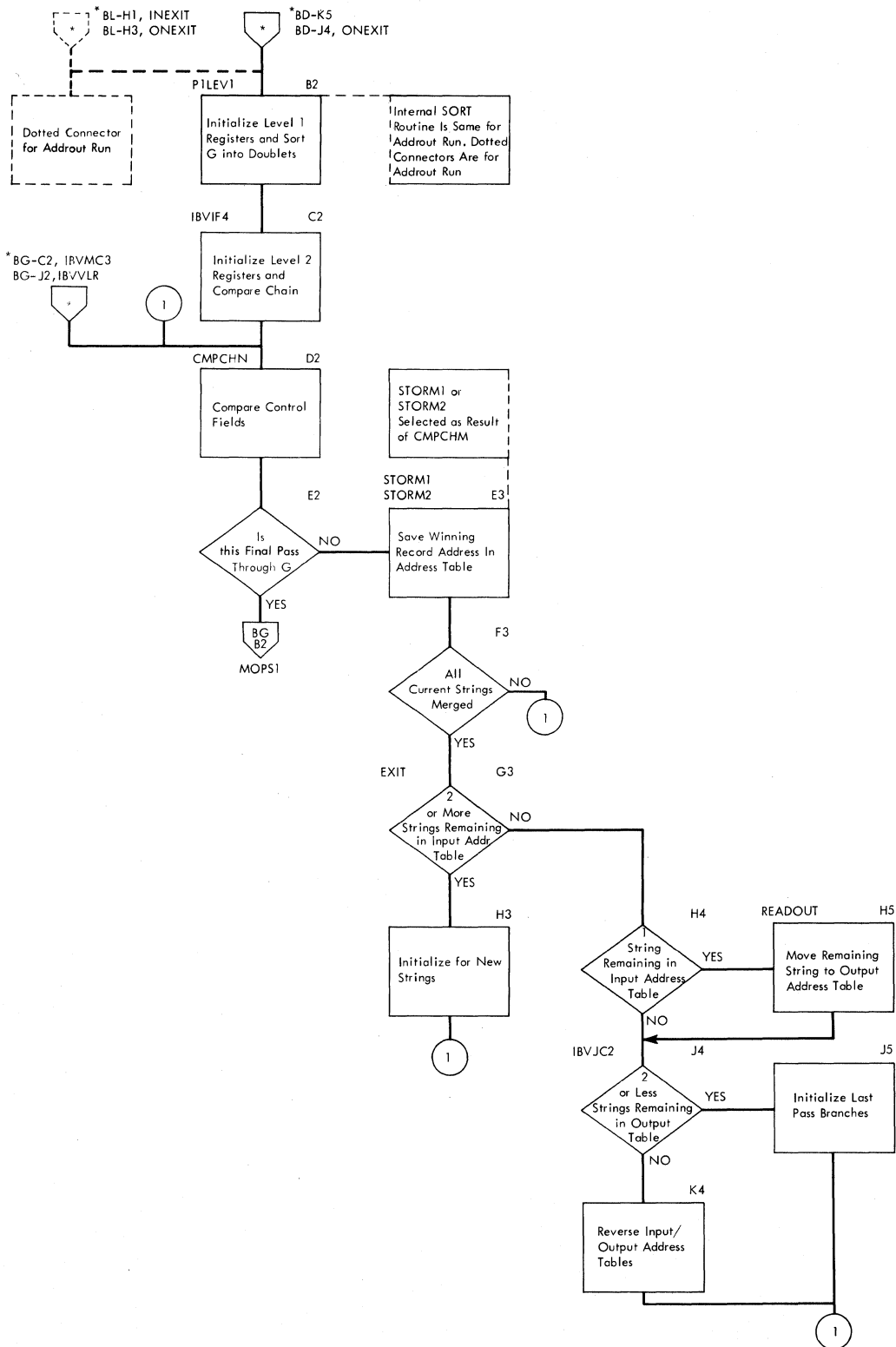


Chart BF. Internal Sort, DSORT103

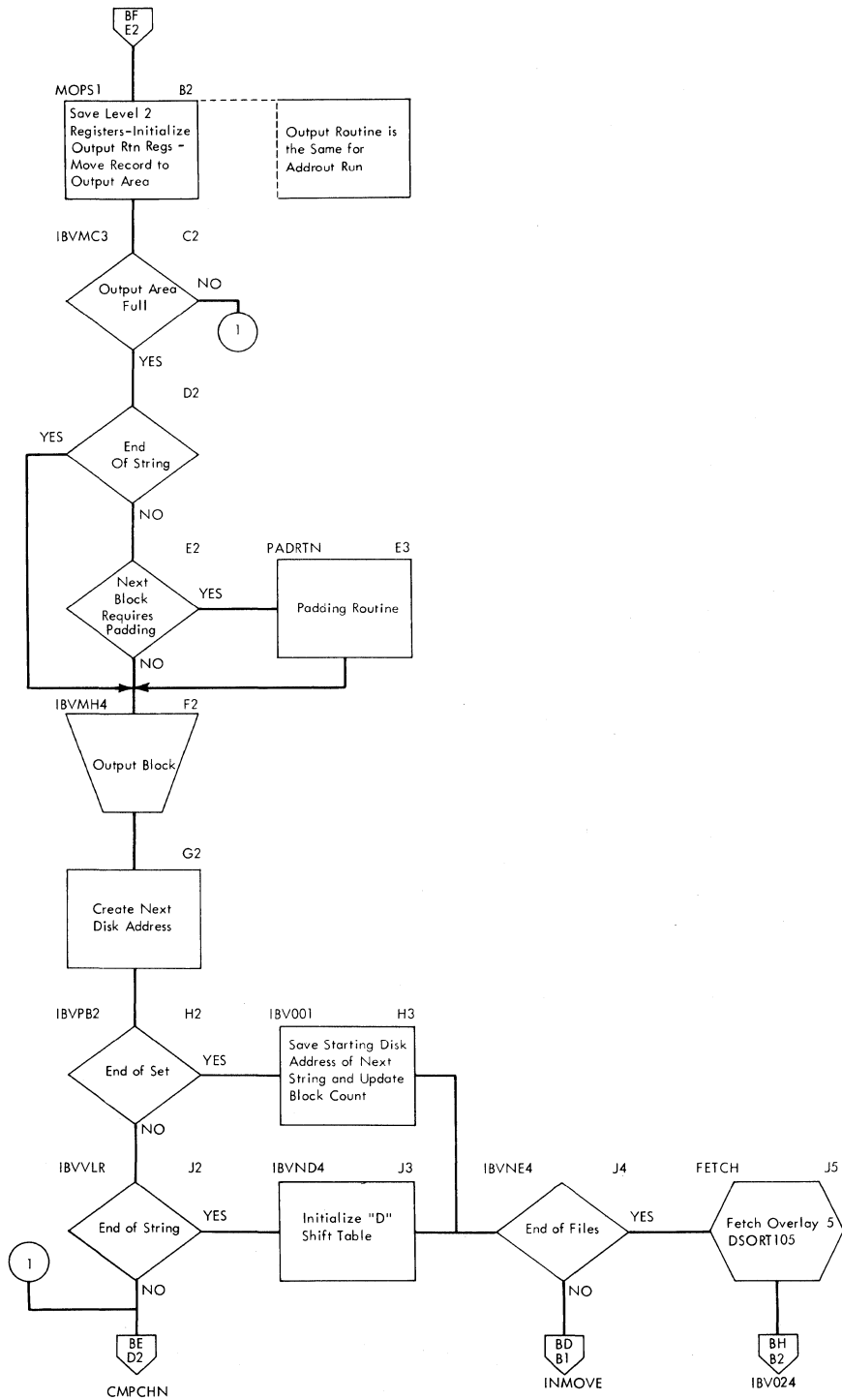


Chart BG. Output Routine, DSORT103

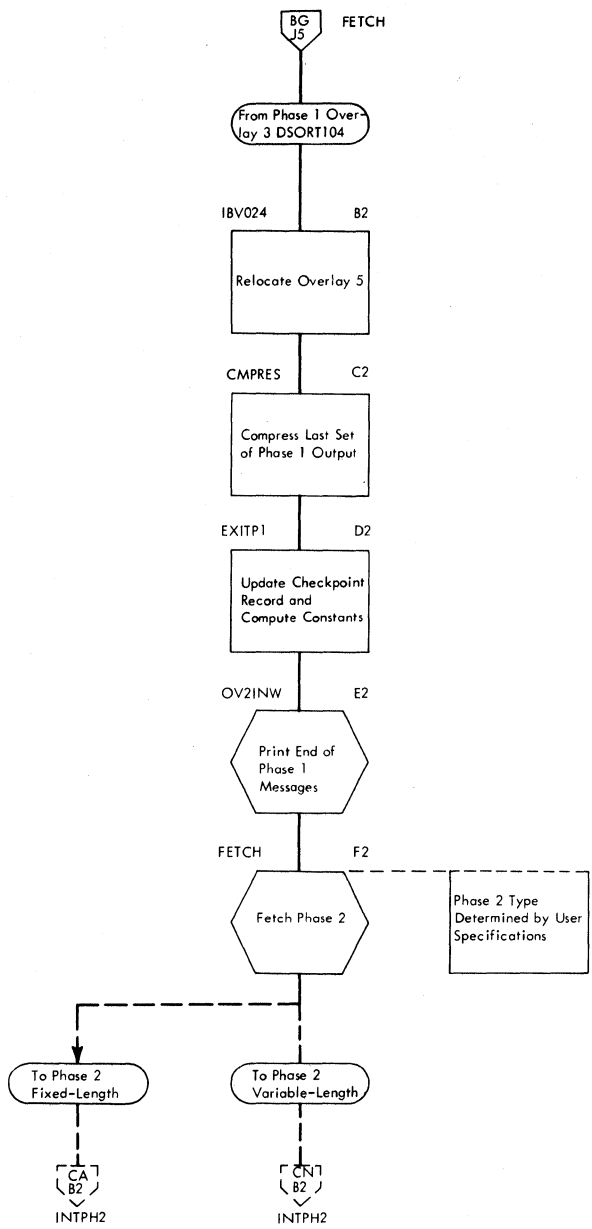


Chart BH. End-of-Phase Routine, DSORT105

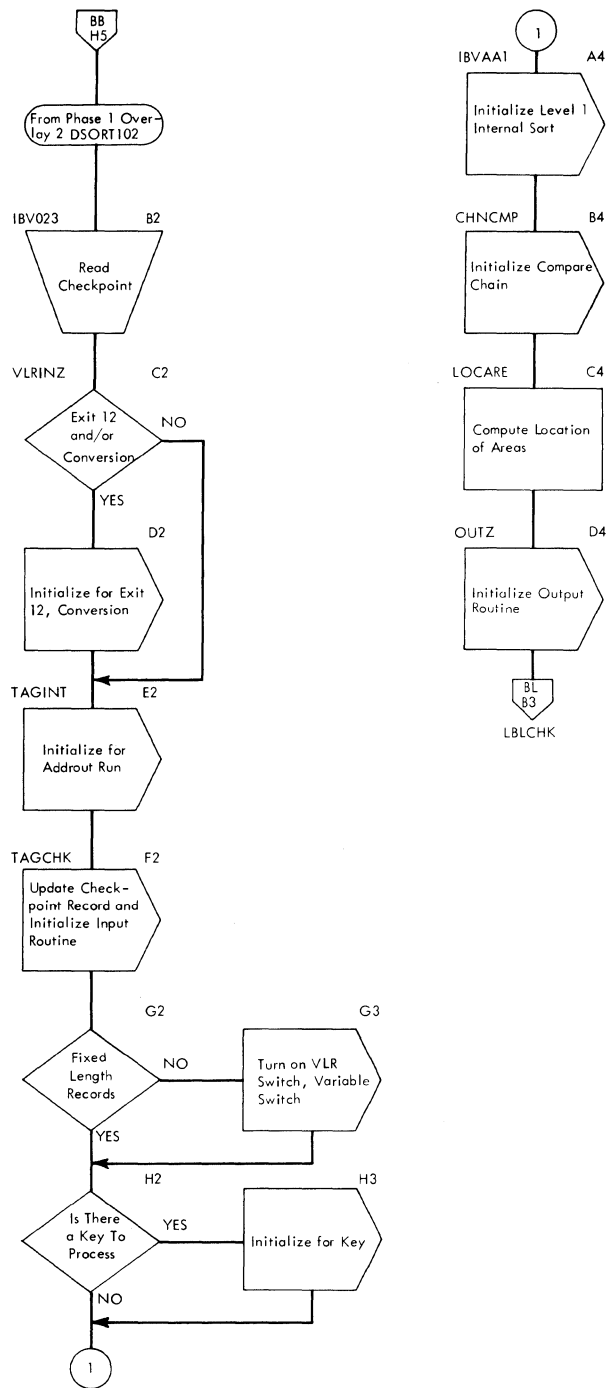


Chart BJ. Initialization for ADDRROUT Run, DSORT104

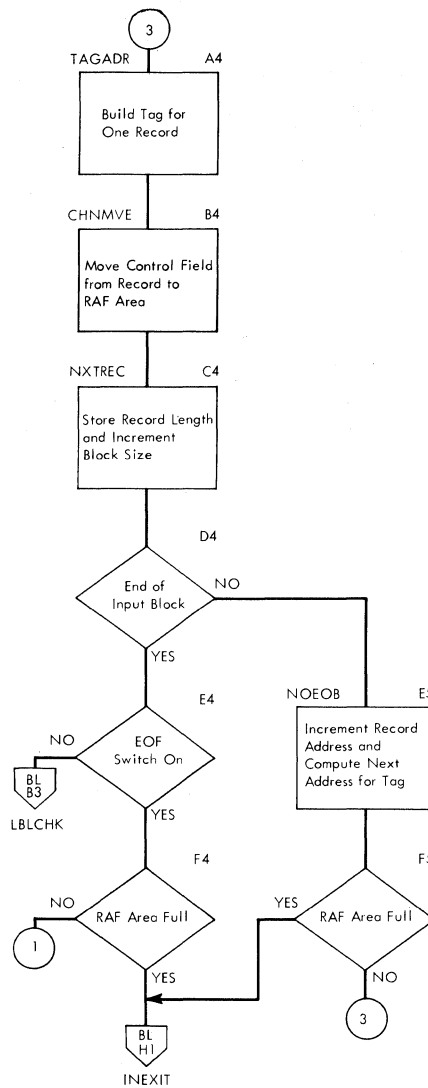
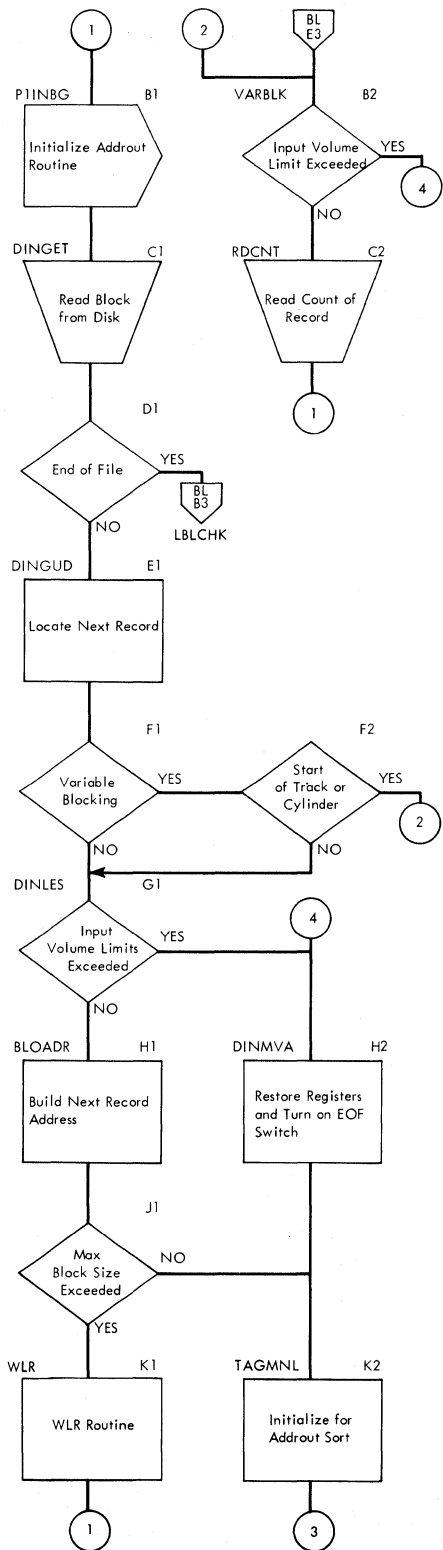


Chart BK. ADDROUT Run Input Routine, DSORT104

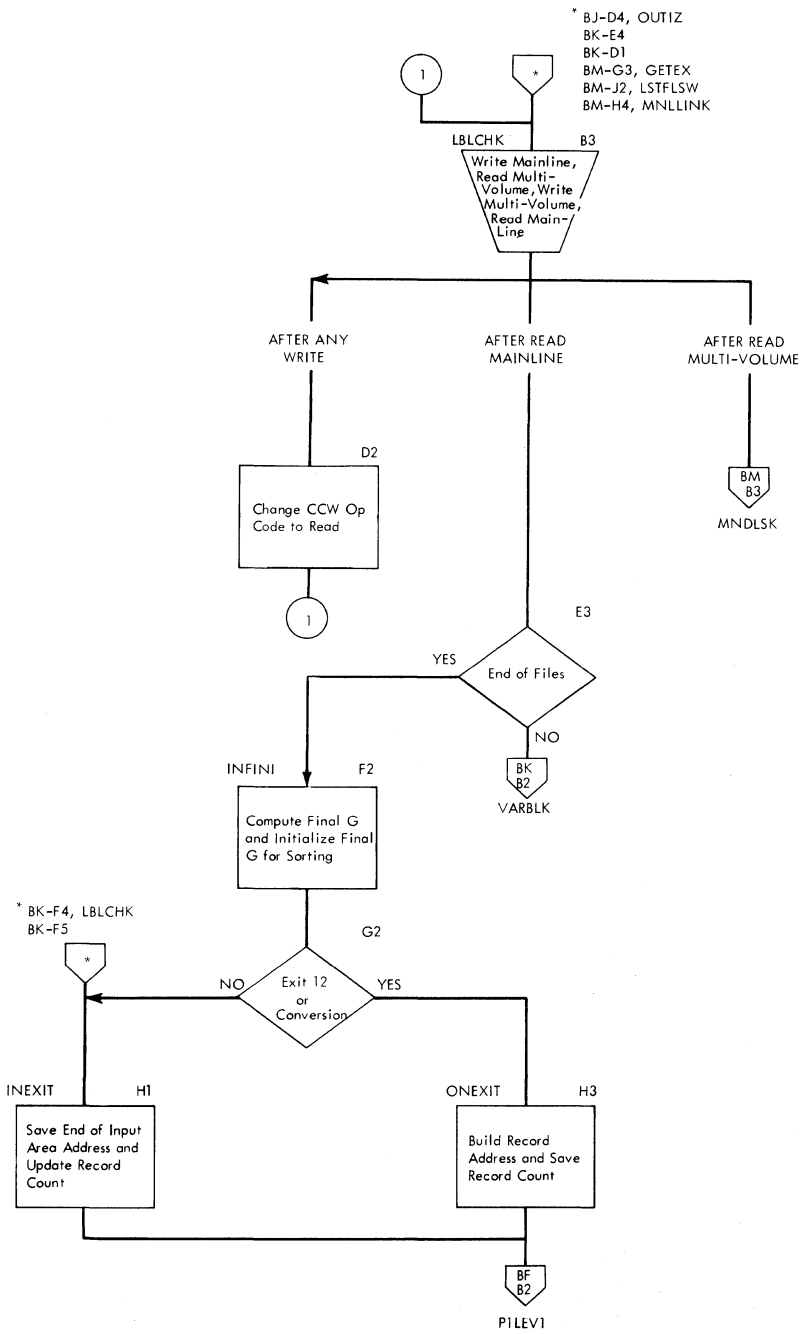


Chart BL. ADDROUT Run Input Routine (Cont'd), DSORT104

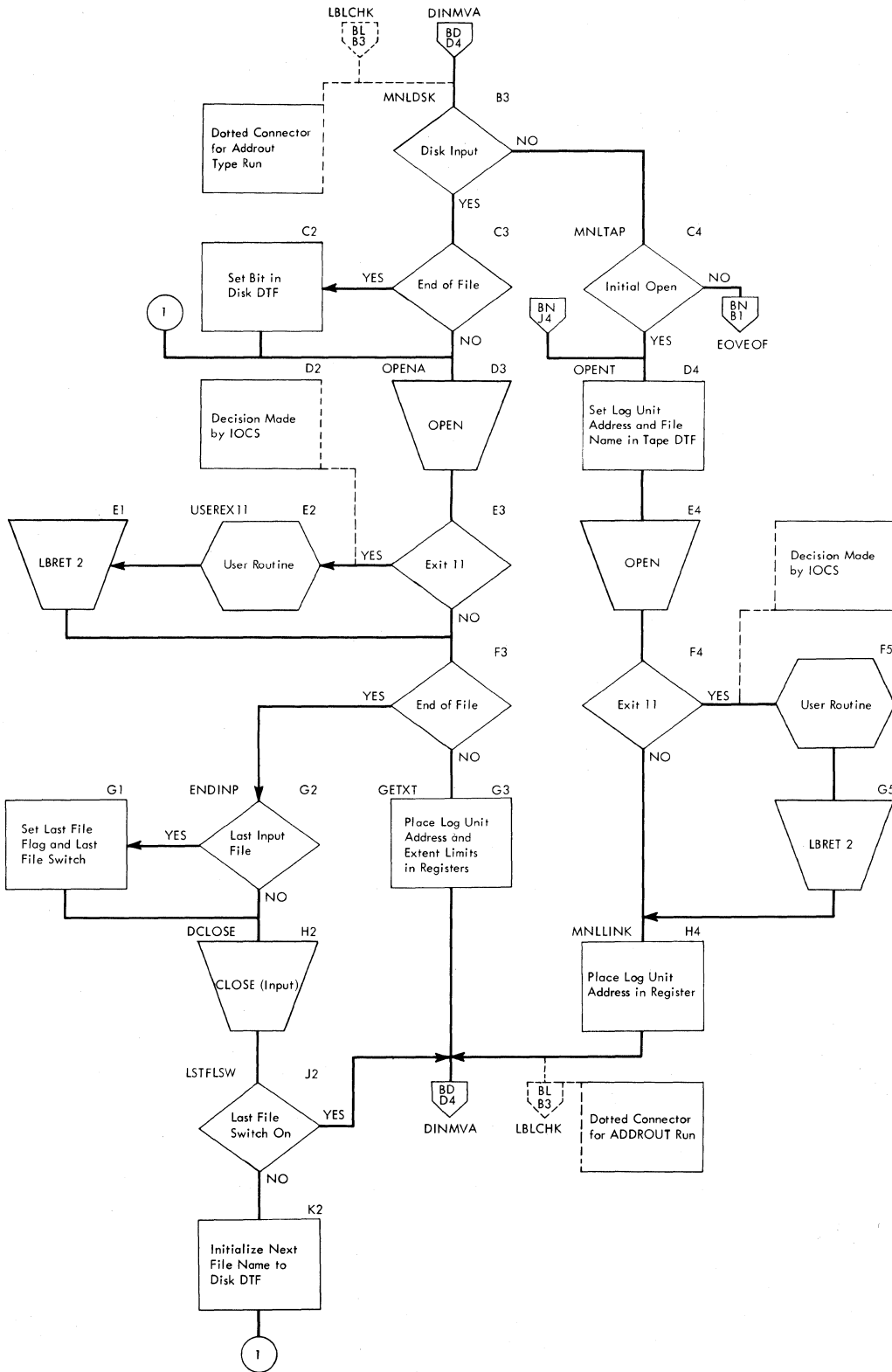


Chart BM. Multi-Volume Routine, DSORT103

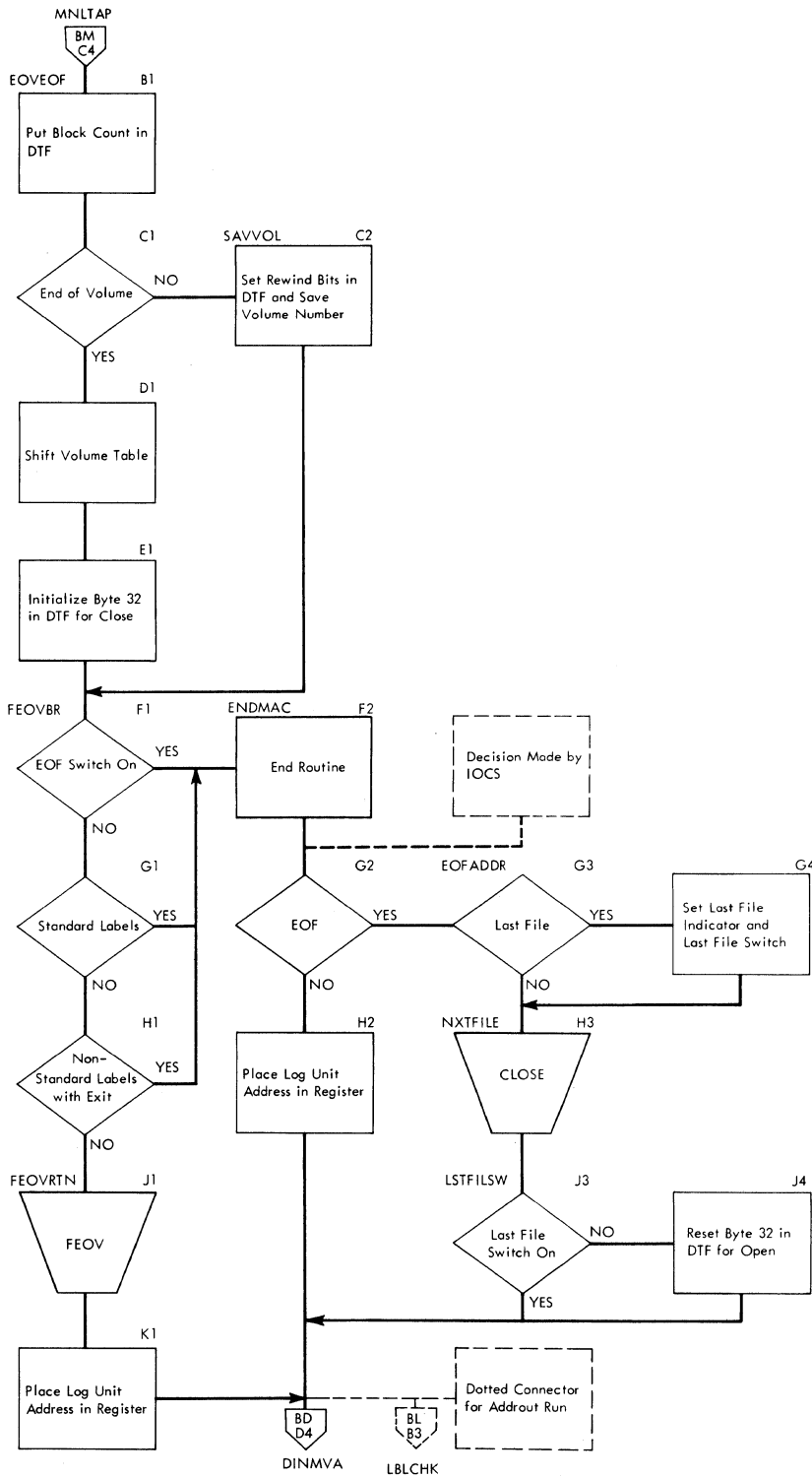


Chart BN. Multi-Volume Routine (Cont'd), DSORT103

DSORT201 - 4-WAY MERGE
 DSORT202 - 7-WAY MERGE

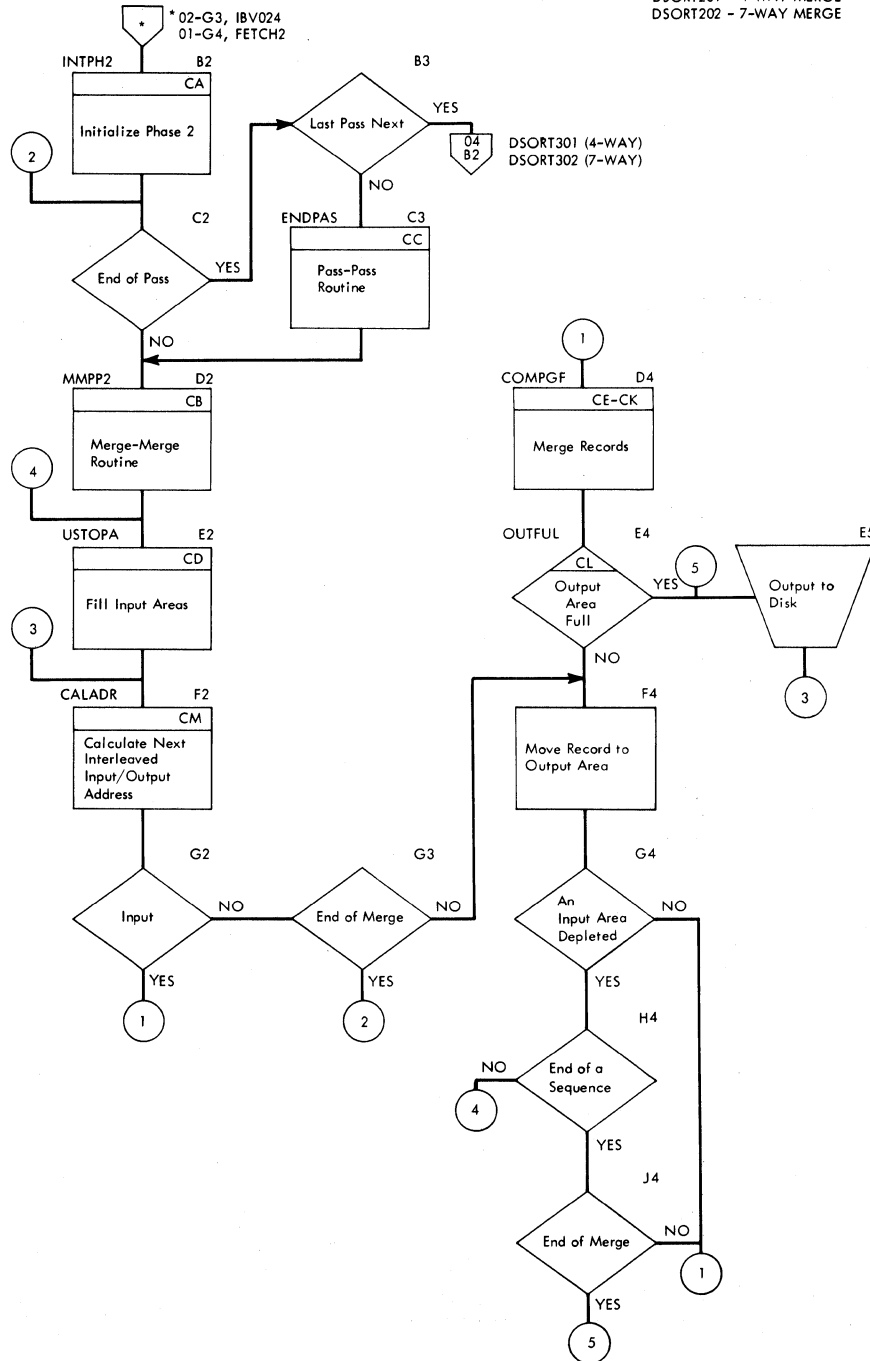


Chart 03. External Sort or Merge, Fixed-Length Records (Phase 2), DSORT201 or DSORT202

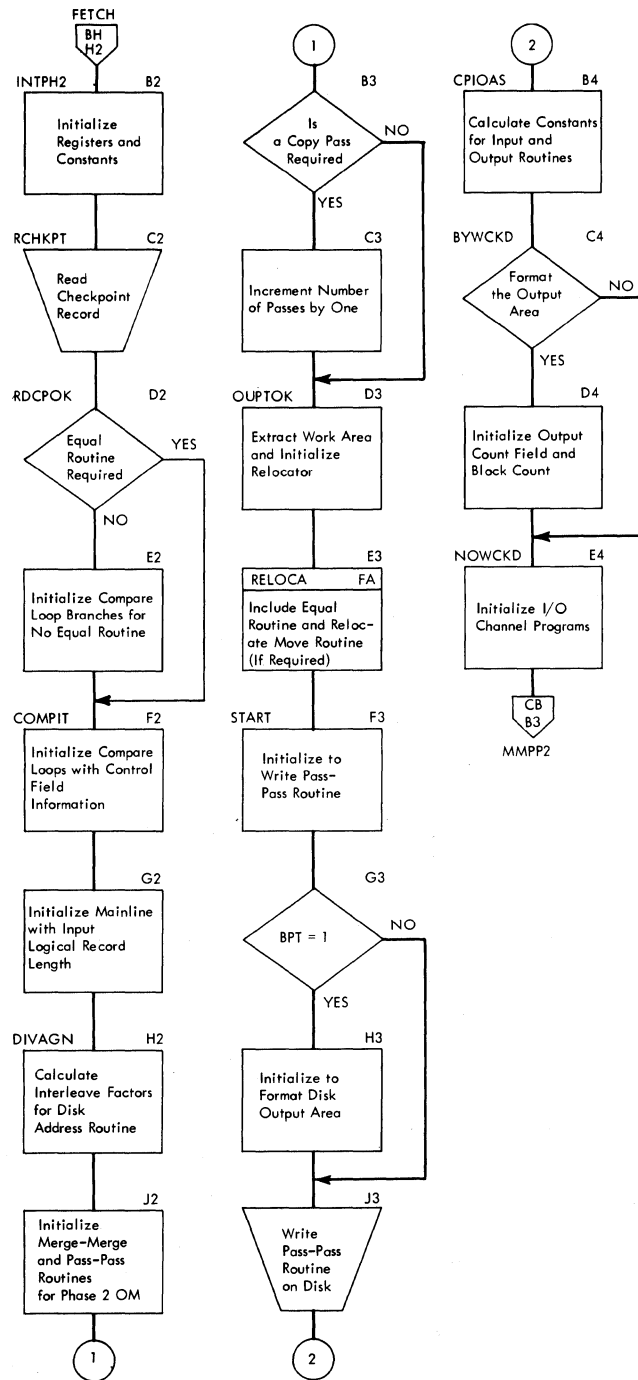


Chart CA. Phase 2 Initialization, Fixed-Length Records

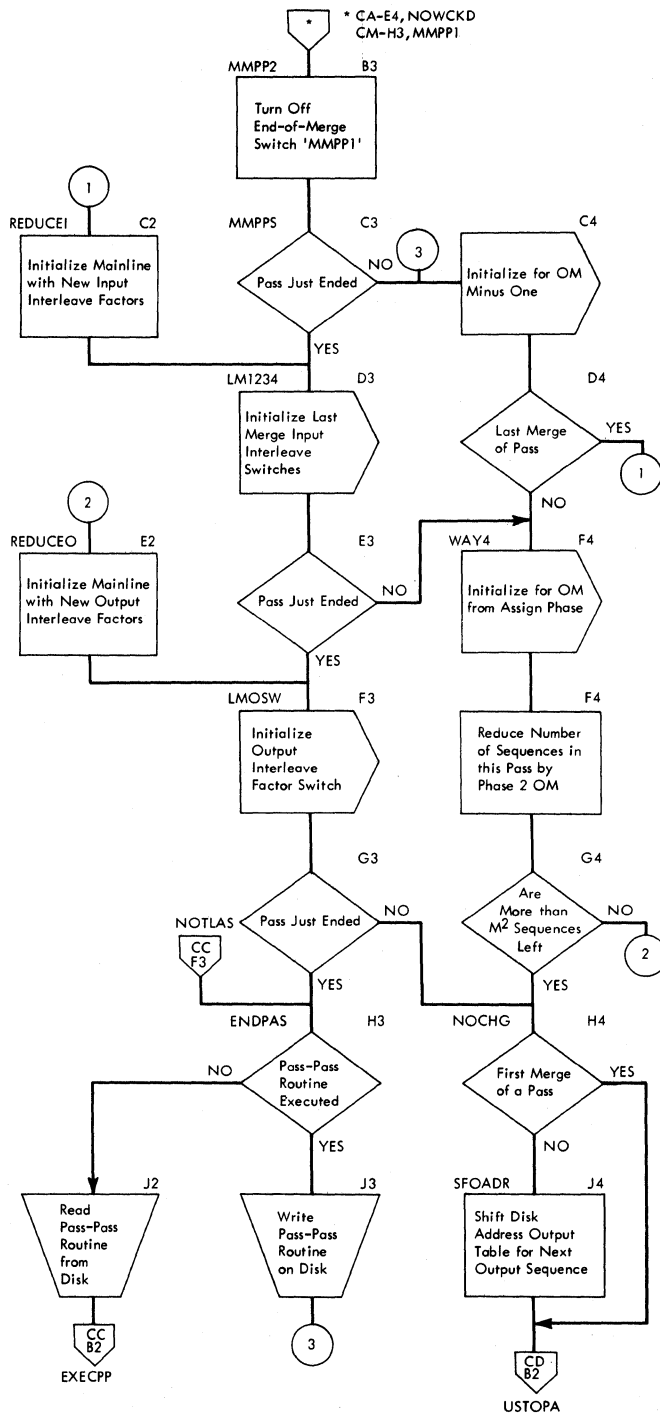


Chart CB. Merge-Merge Routine, Fixed-Length Records

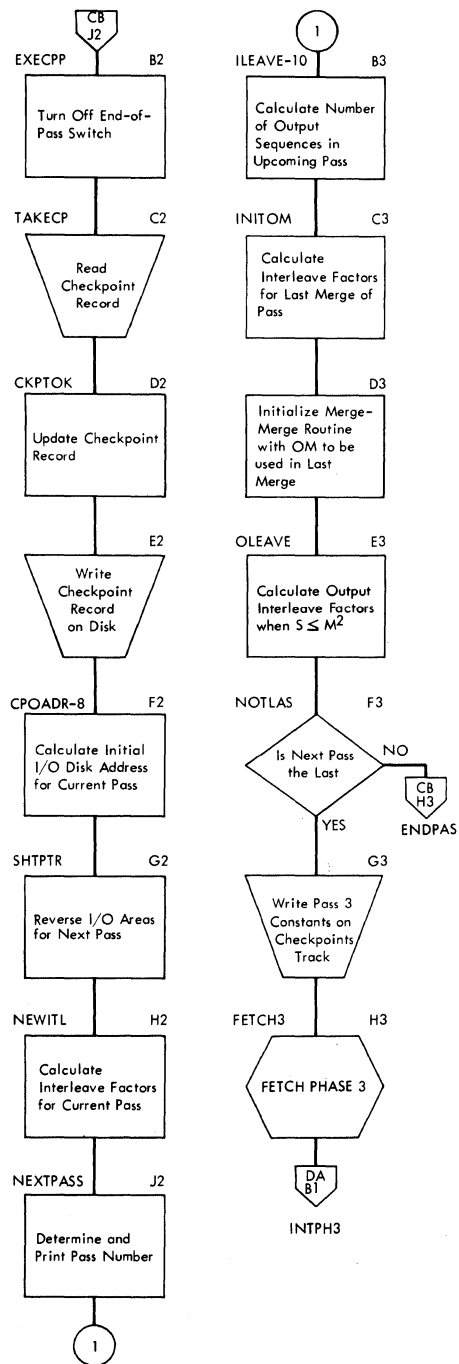


Chart CC. Pass-Pass Routine, Fixed-Length Records

*CB-J4 CH-J3
 CE-J4 CJ-H3
 CF-J4 CK-H2
 CG-J4 CK-H4

NOTE: USTOPA, FILLA, and GETA labels apply to 1-way merge. For 2-way to 7-way merges, label suffixes are B to G, respectively. Similarly, LM1 and BYPAS1+4 become LM2 to LM6 and BYPAS2+4 to BYPAS7+4, respectively.

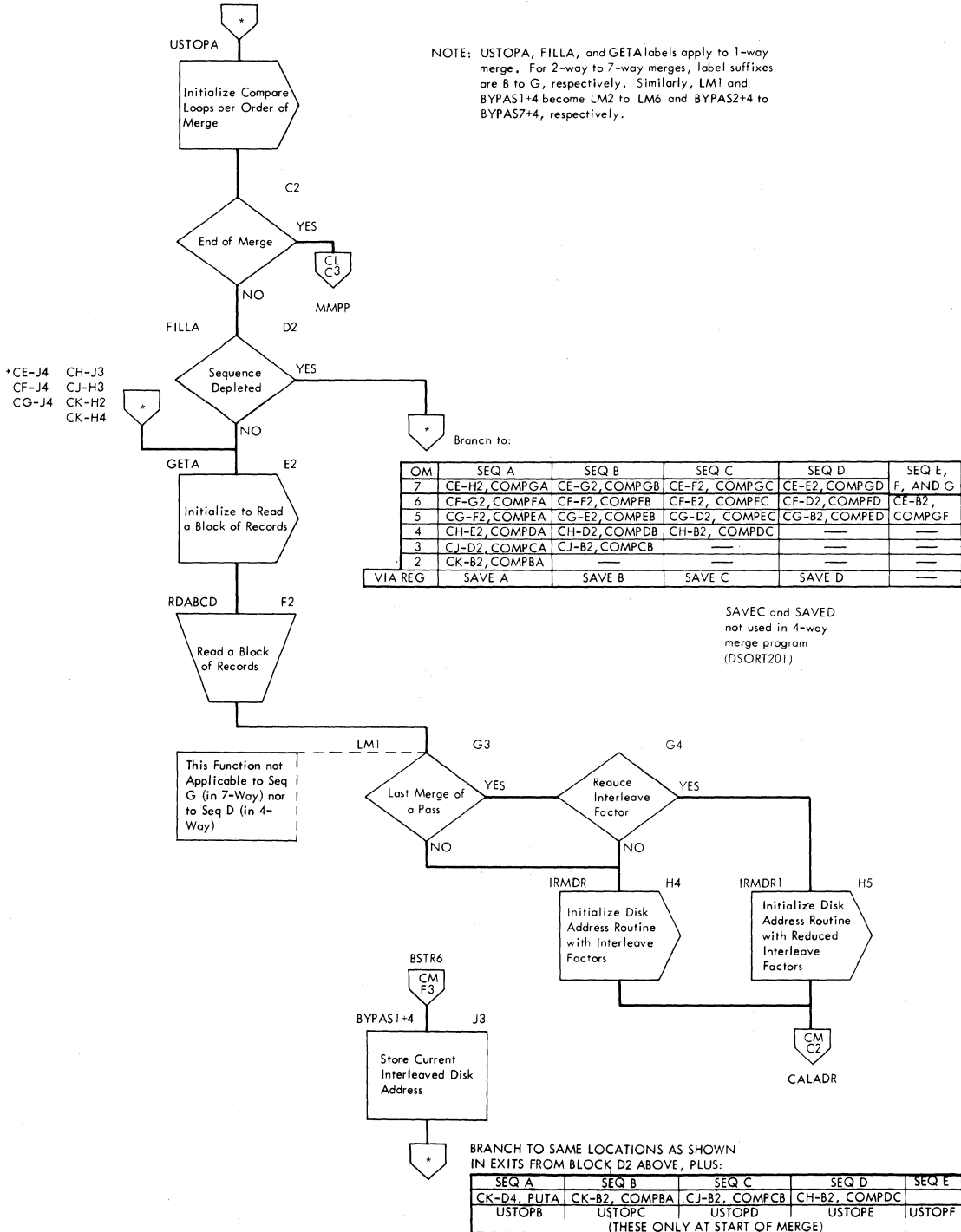


Chart CD. Input Routine, Fixed-Length Records

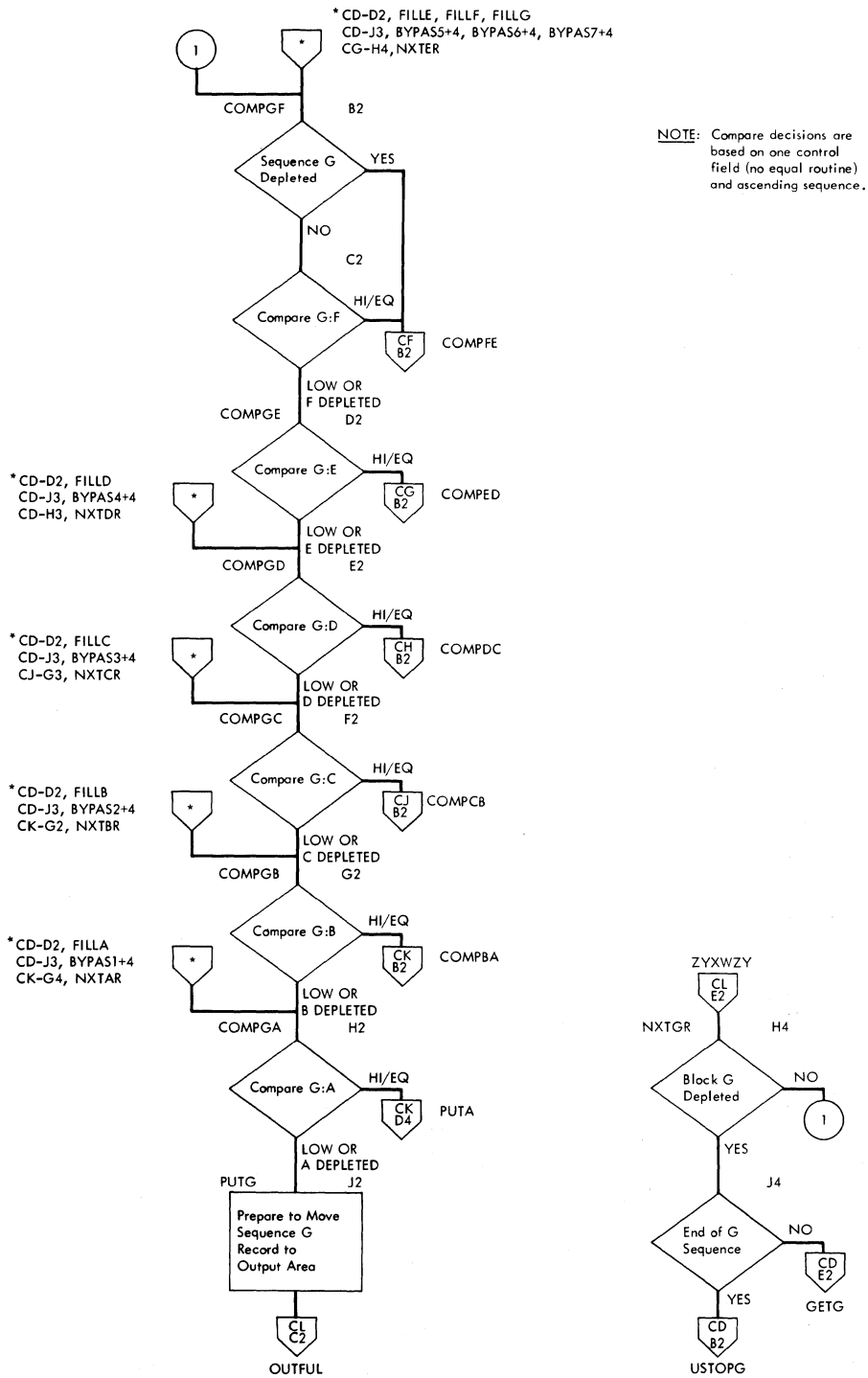
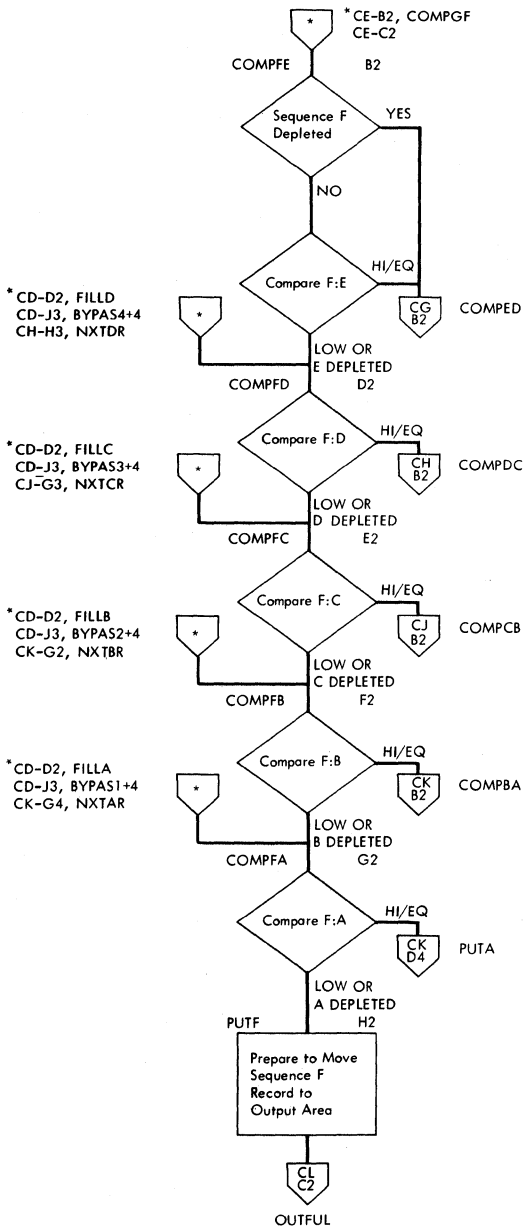


Chart CE. Sequence G Compare Loop, Fixed-Length Records



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

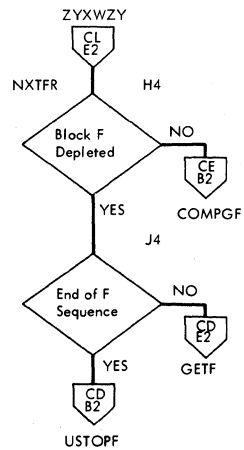
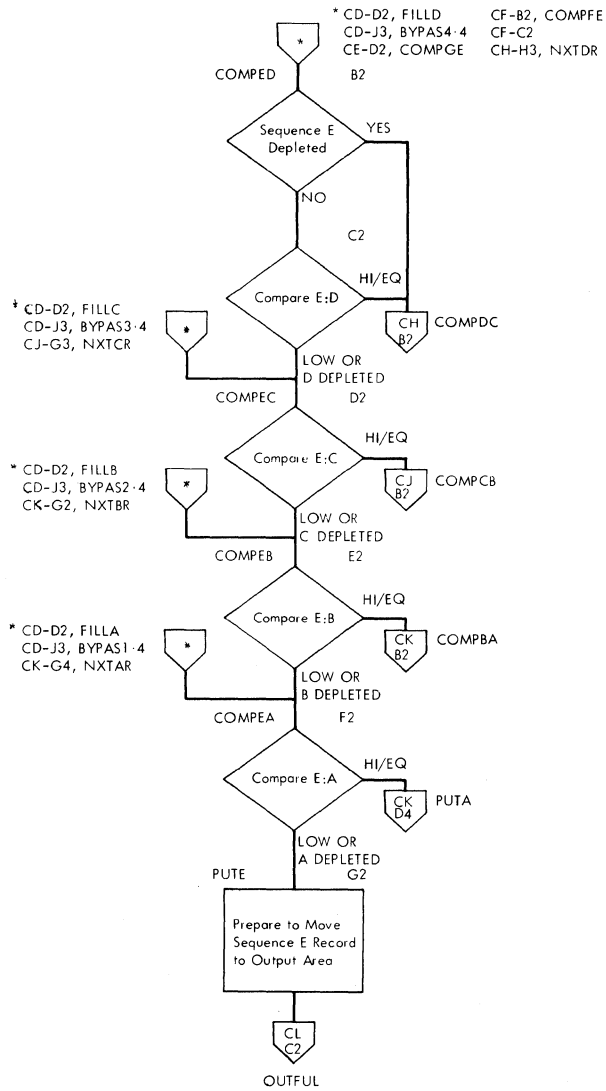


Chart CF. Sequence F Compare Loop, Fixed-Length Records



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

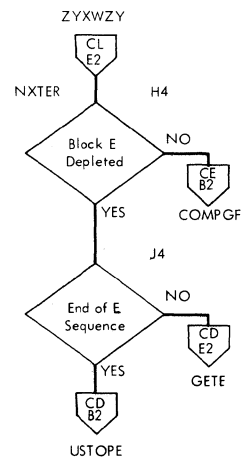


Chart CG. Sequence E Compare Loop, Fixed-Length Records

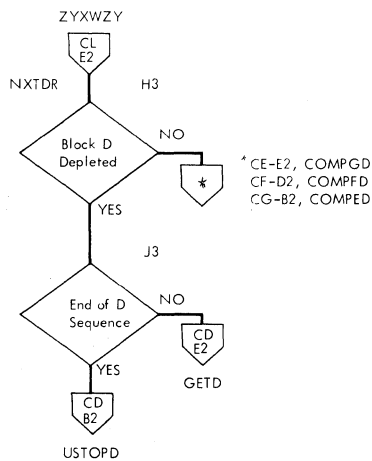
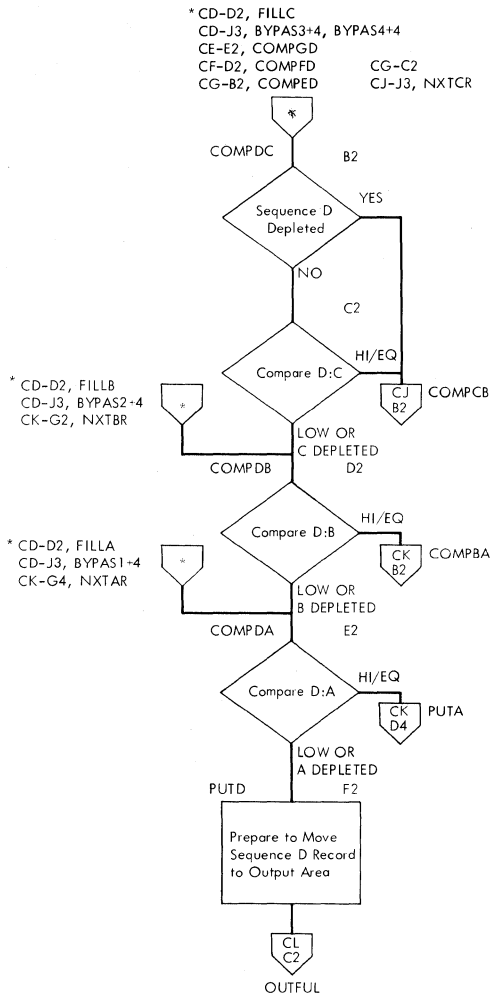
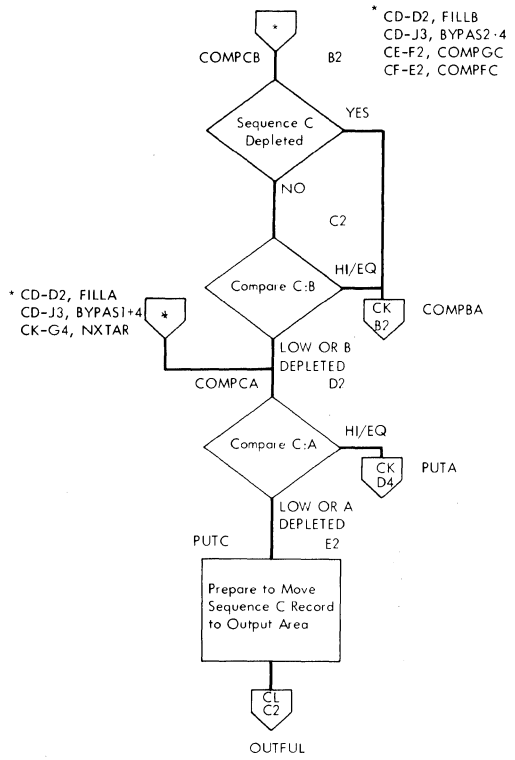


Chart CH. Sequence D Compare Loop, Fixed-Length Records



CG-D2, COMPEC
 CH-B2, COMPCD
 CH-C2
 CK-G2, NXTBR

NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

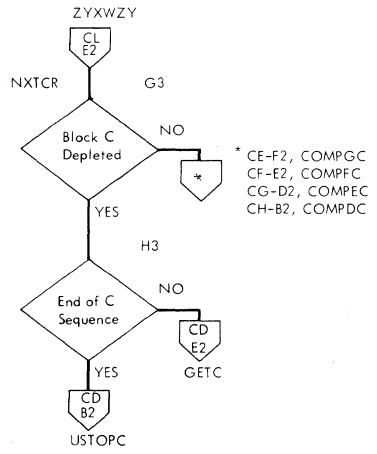


Chart CJ. Sequence C Compare Loop, Fixed-Length Records

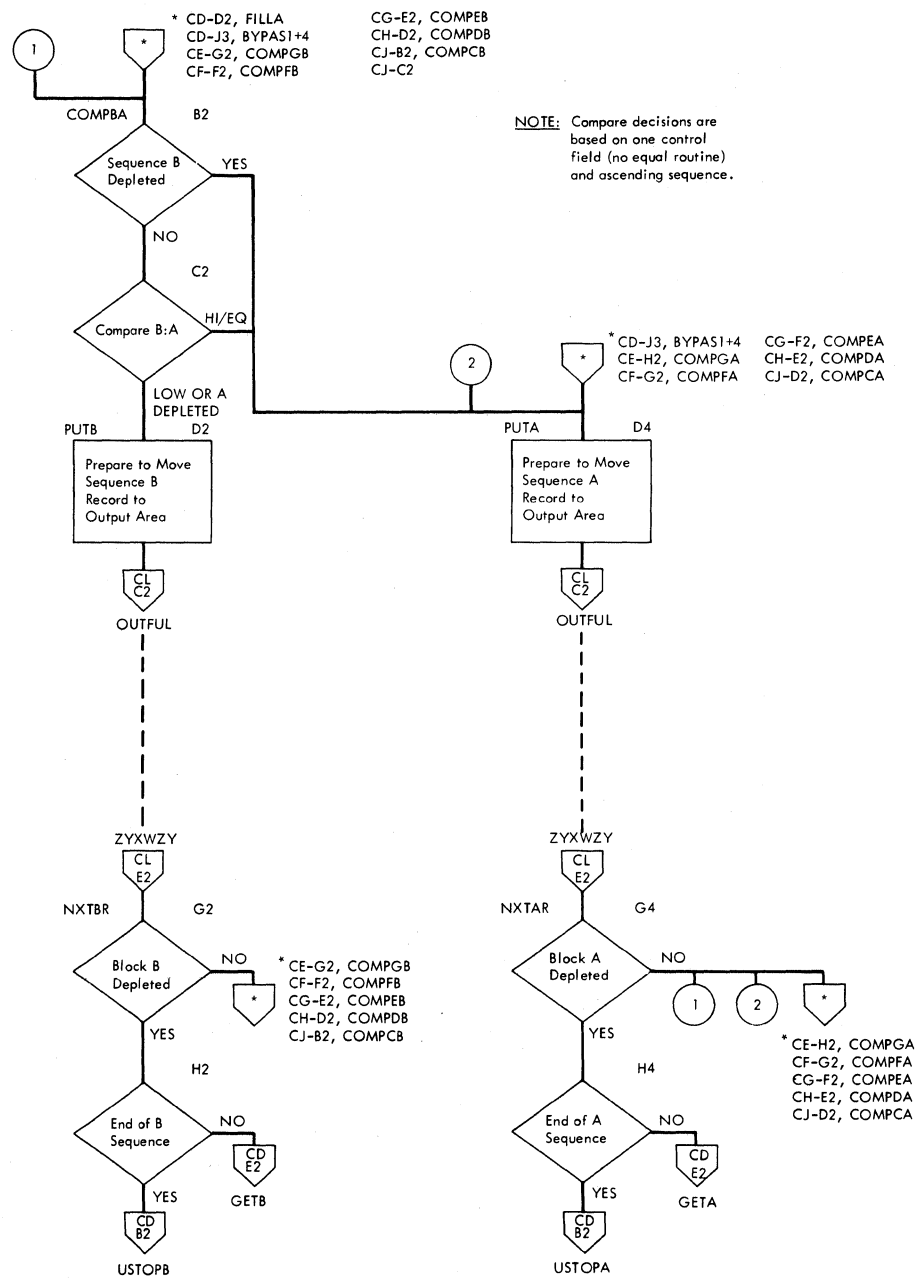


Chart CK. Sequence B Compare Loop, Fixed-Length Records

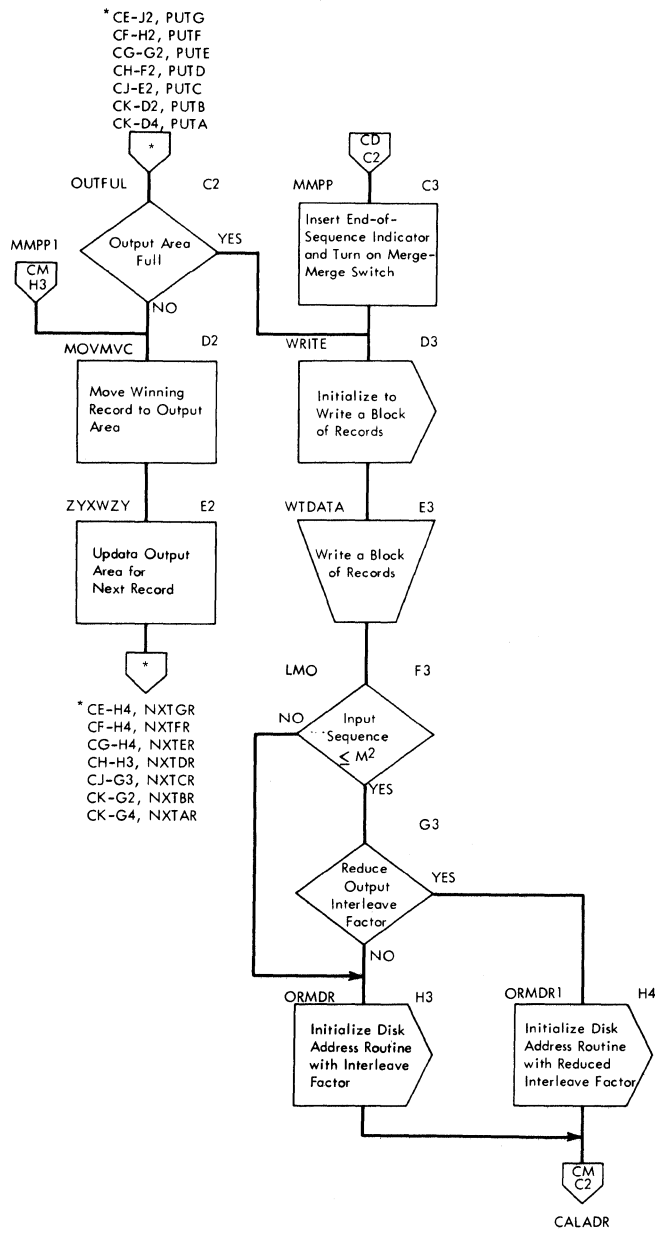


Chart CL. Output Routine, Fixed-Length Records

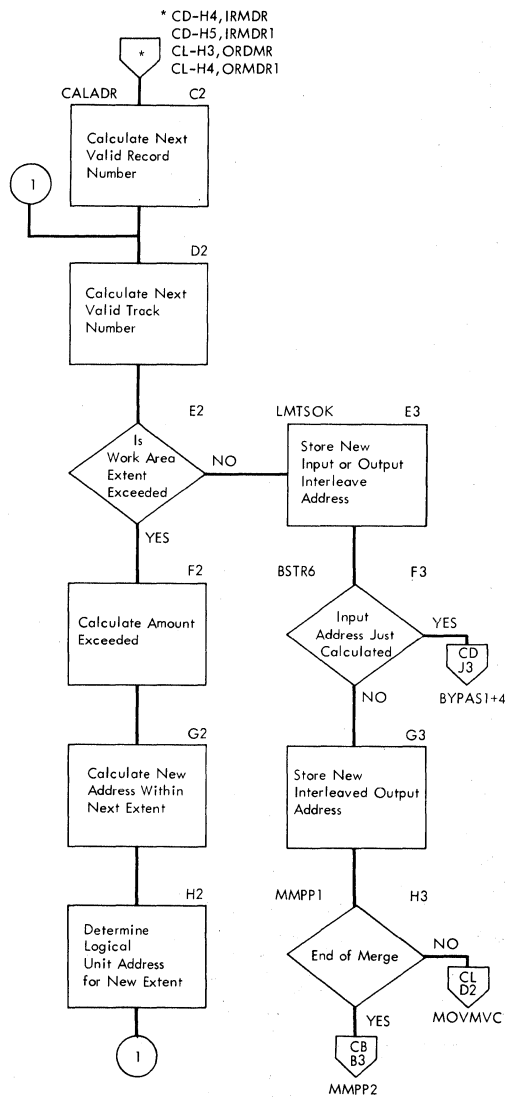


Chart CM. Calculate Interleave Disk Address, Fixed-Length Records

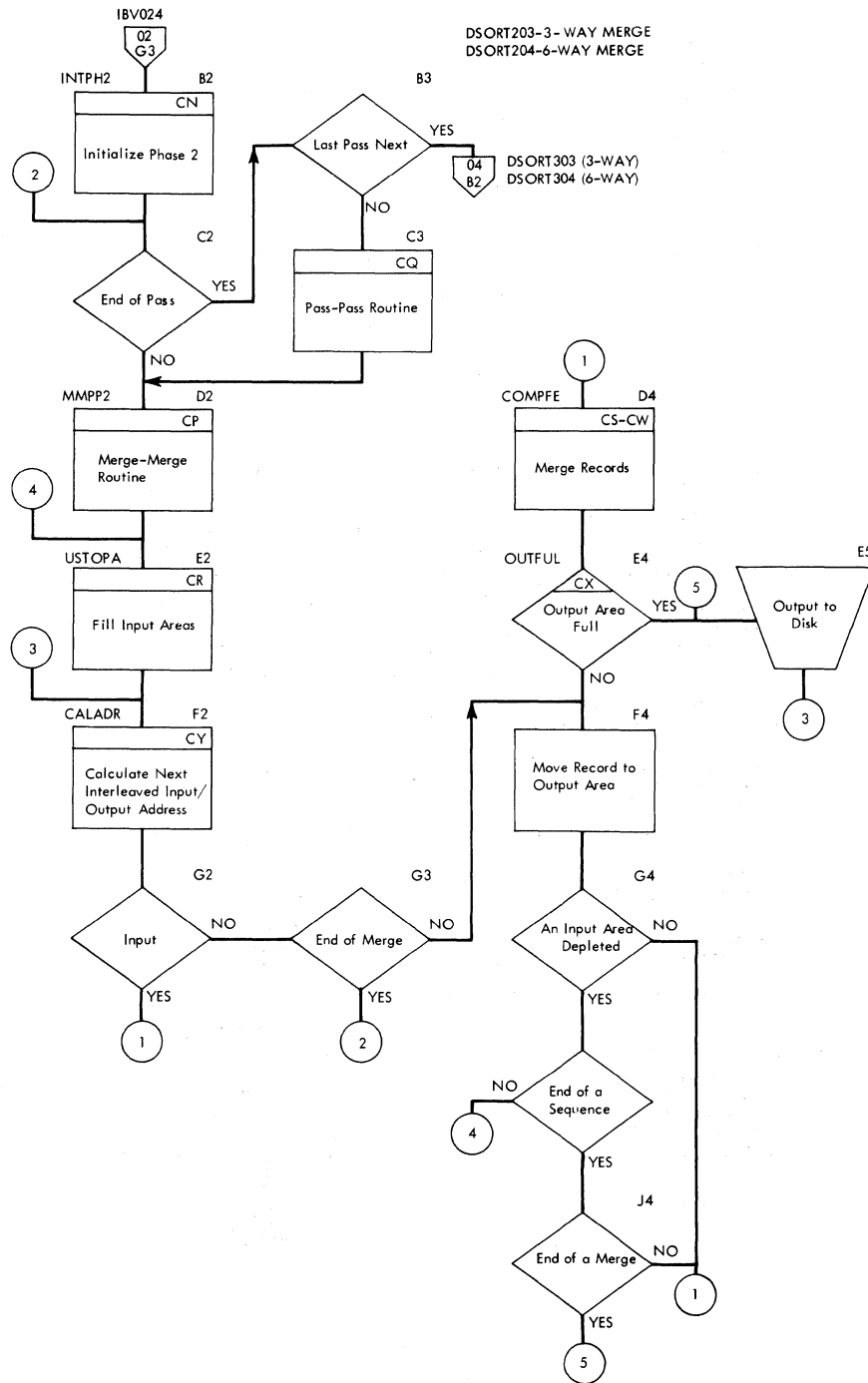


Chart 03. External Sort or Merge, Variable-Length Records (Phase 2), DSORT203 or DSORT204

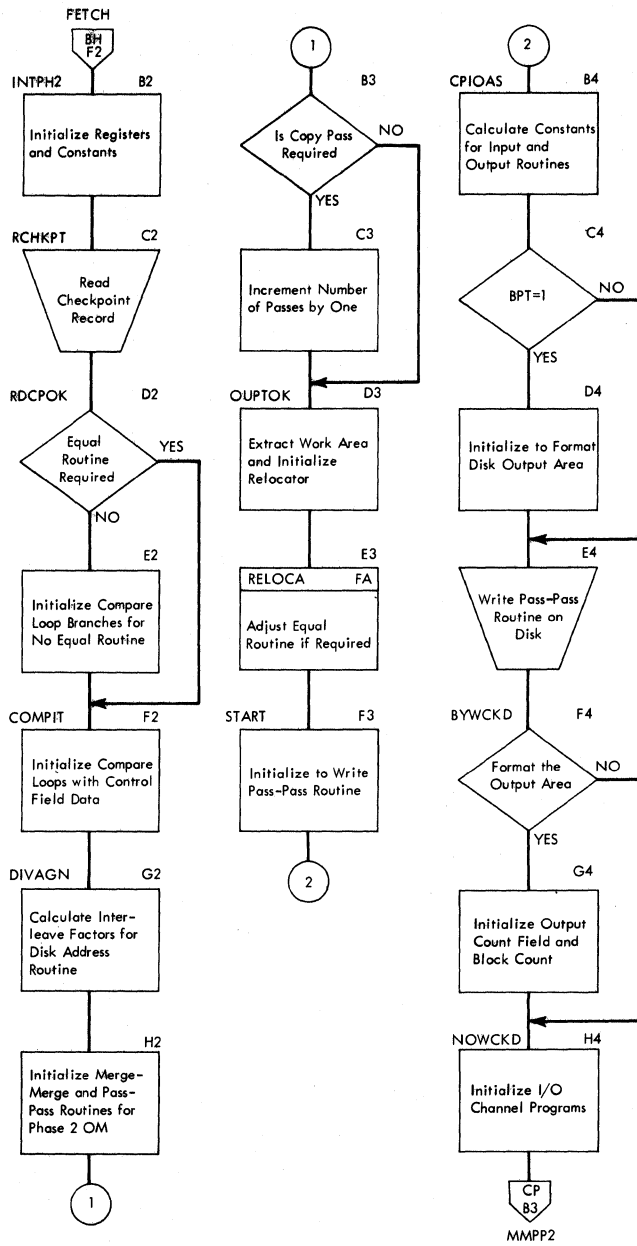


Chart CN. Phase 2 Initialization, Variable-Length Records

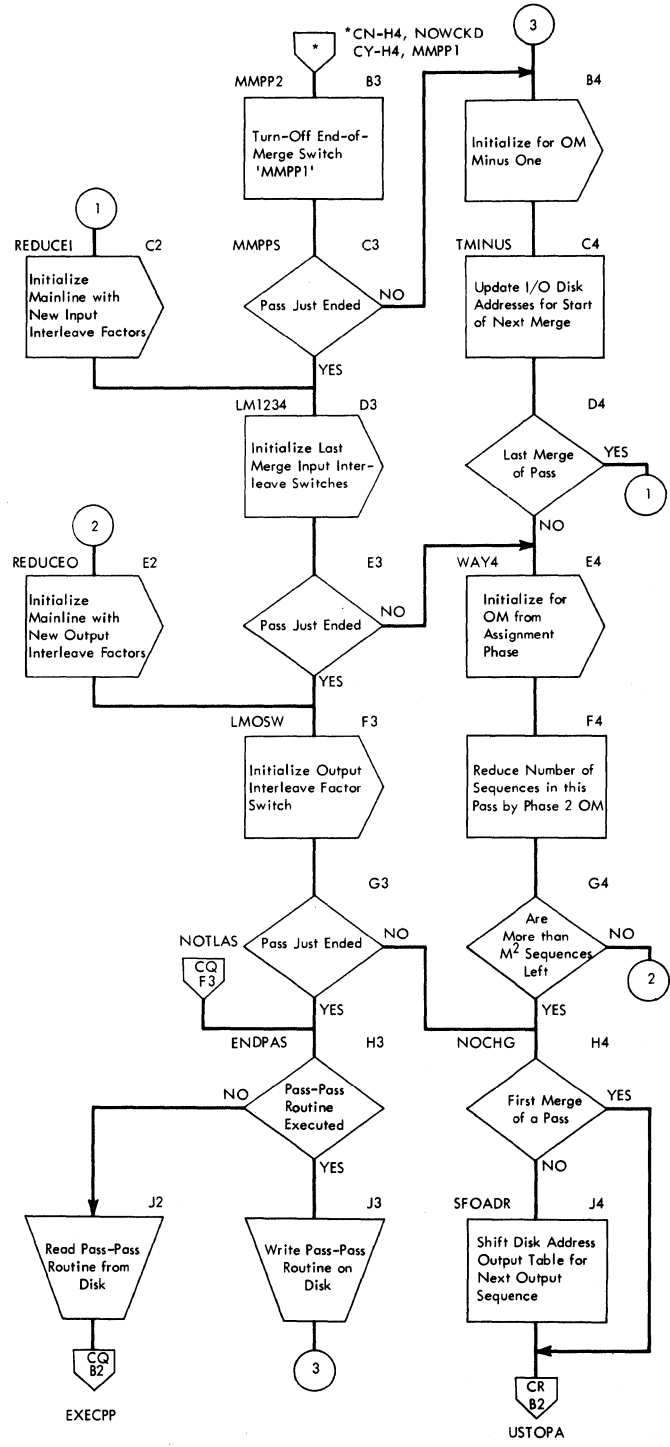


Chart CP. Merge-Merge Routine, Variable-Length Records

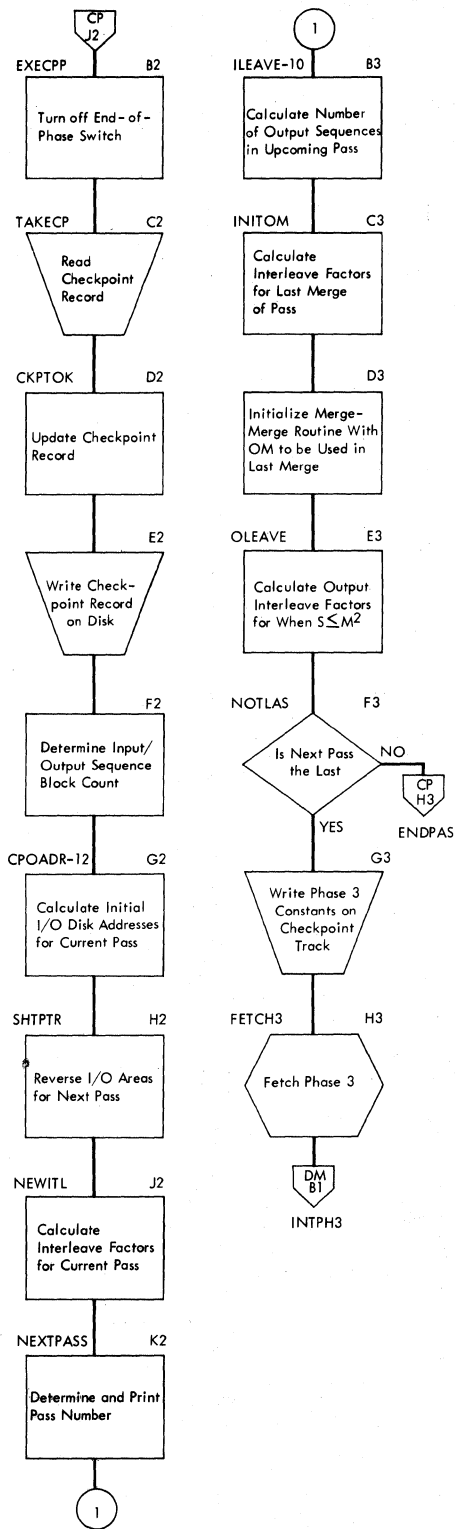


Chart CQ. Pass-Pass Routine, Variable-Length Records

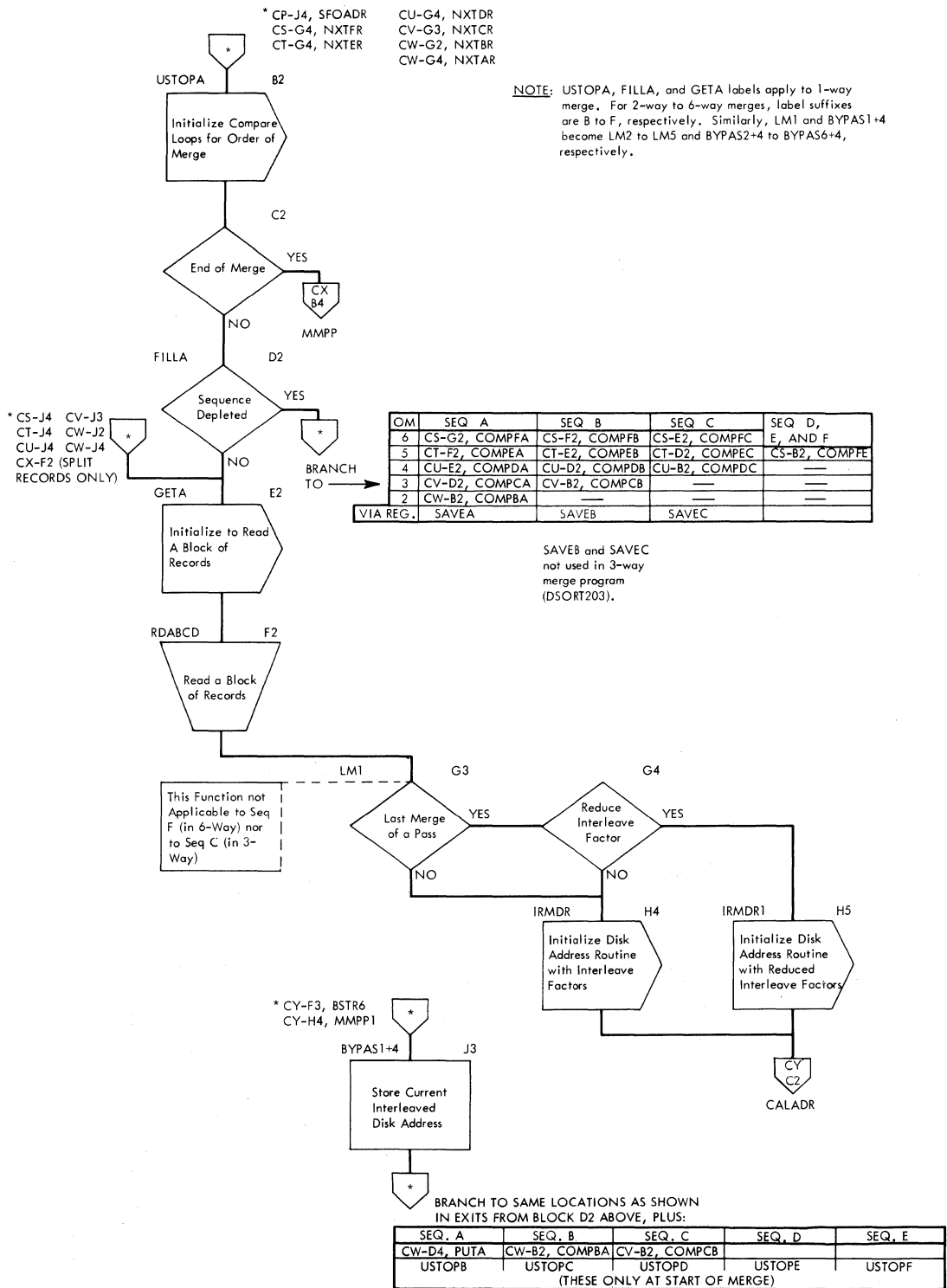
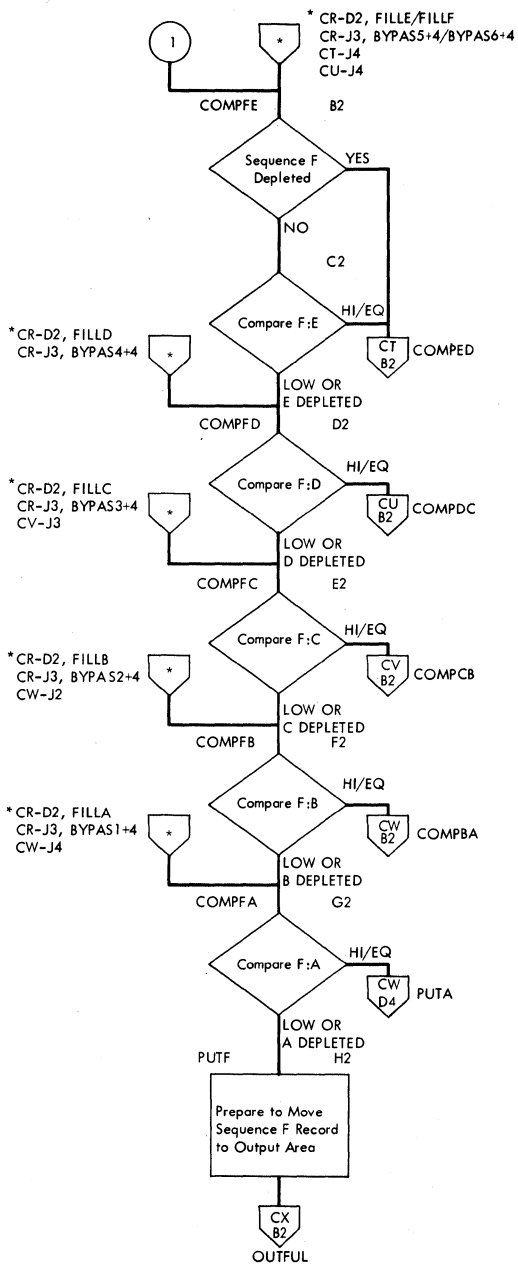


Chart CR. Input Routine, Variable-Length Records



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

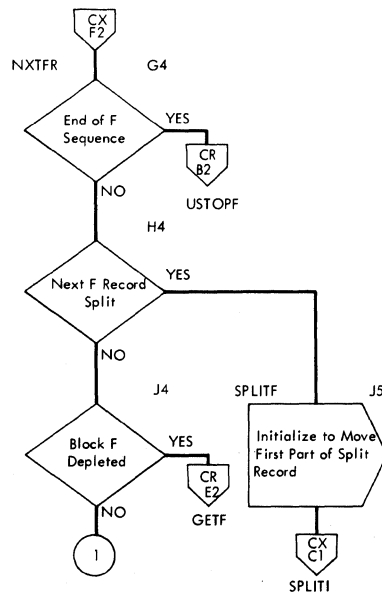
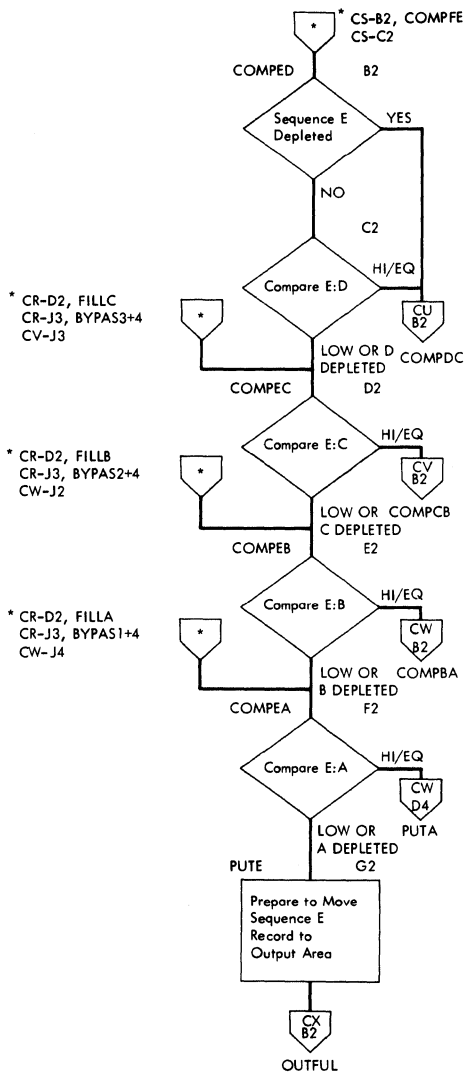


Chart CS. Sequence F Compare Loop, Variable-Length Records



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

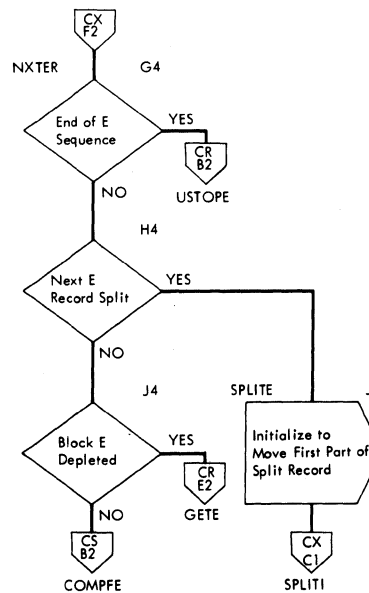
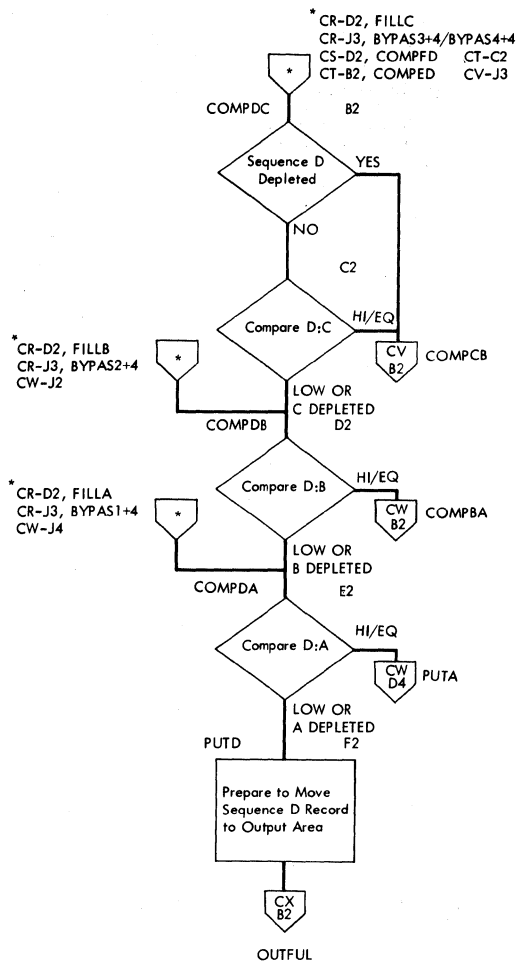


Chart CT. Sequence E Compare Loop, Variable-Length Records



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

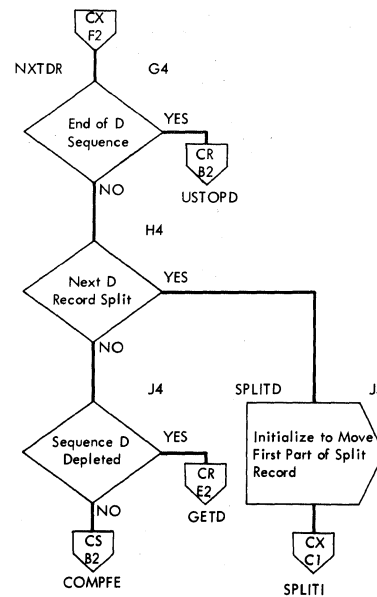


Chart CU. Sequence D Compare Loop, Variable-Length Records

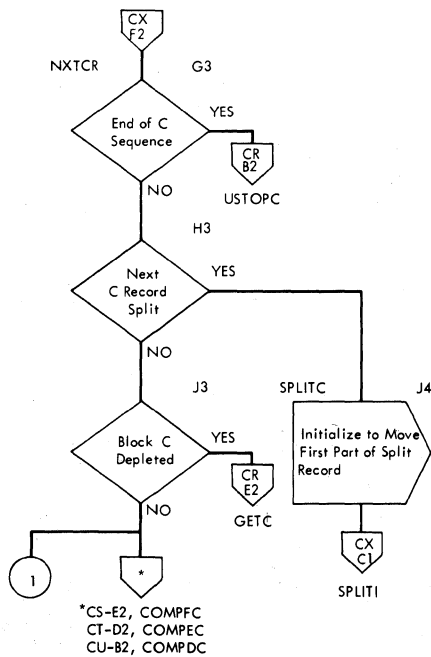
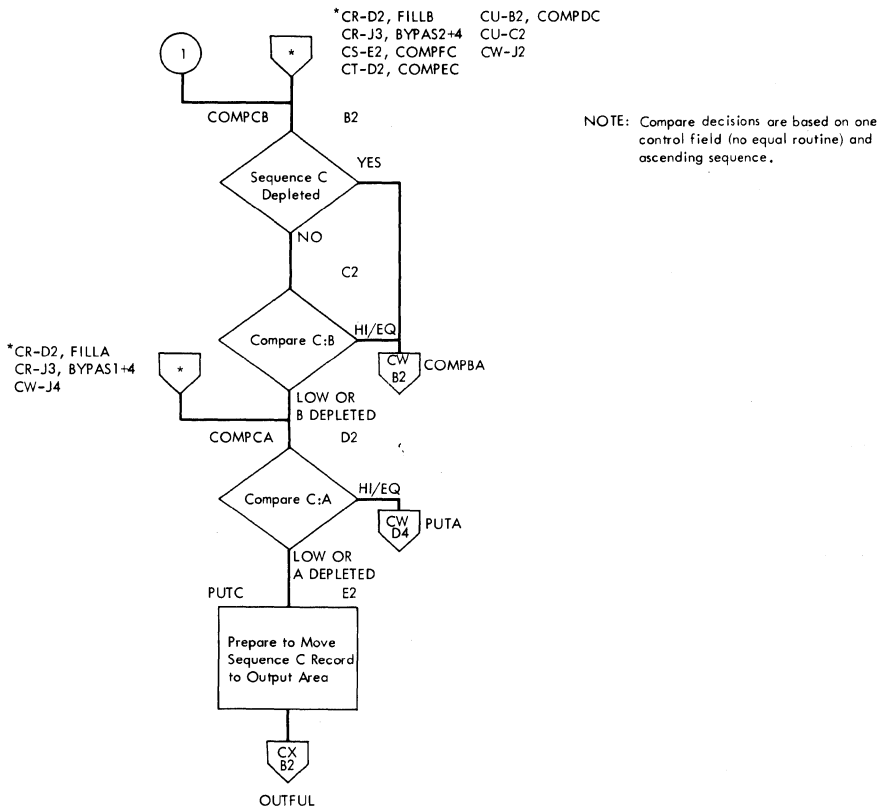


Chart CV. Sequence C Compare Loop, Variable-Length Records

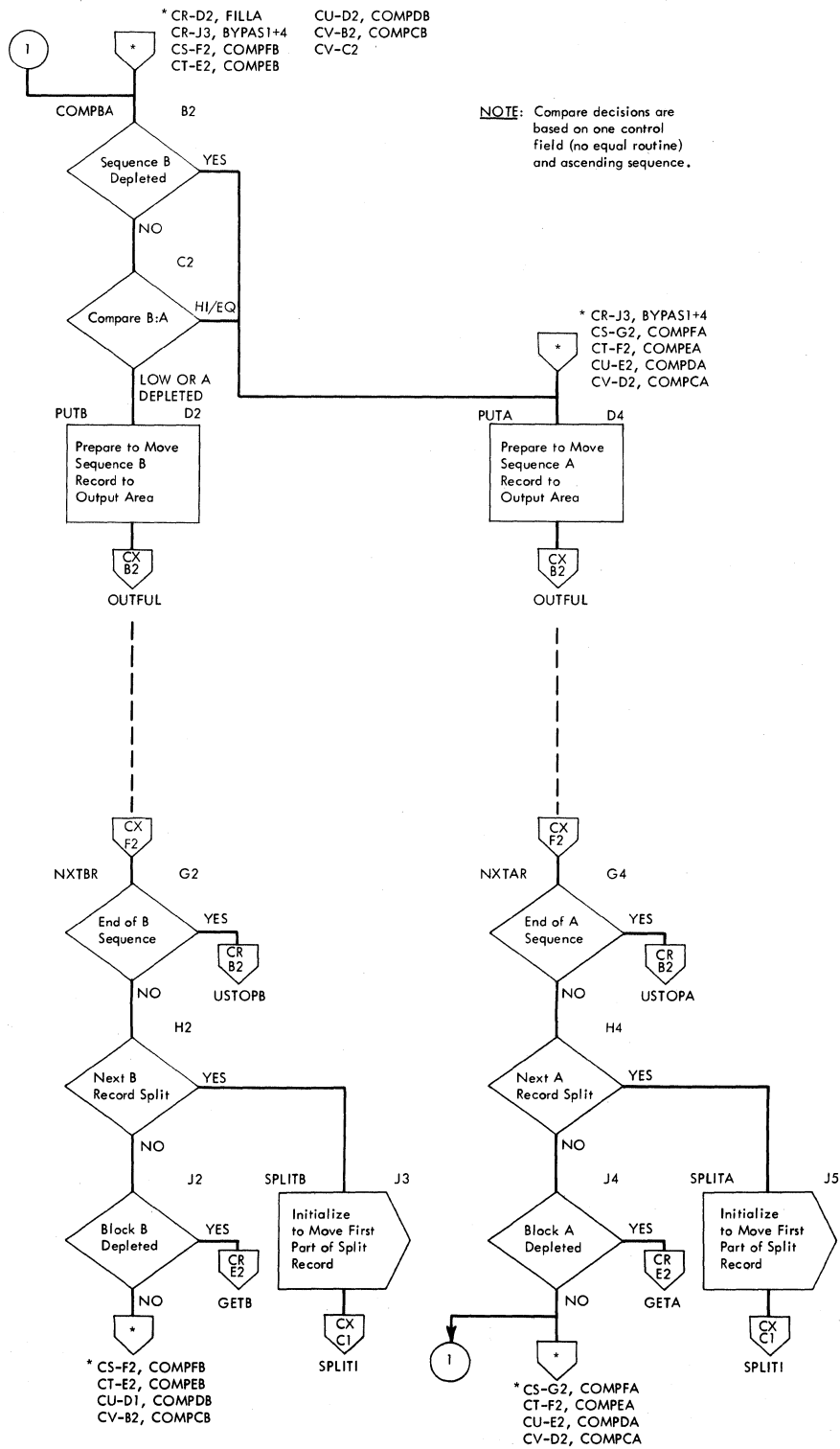


Chart CW. Sequence B Compare Loop, Variable-Length Records

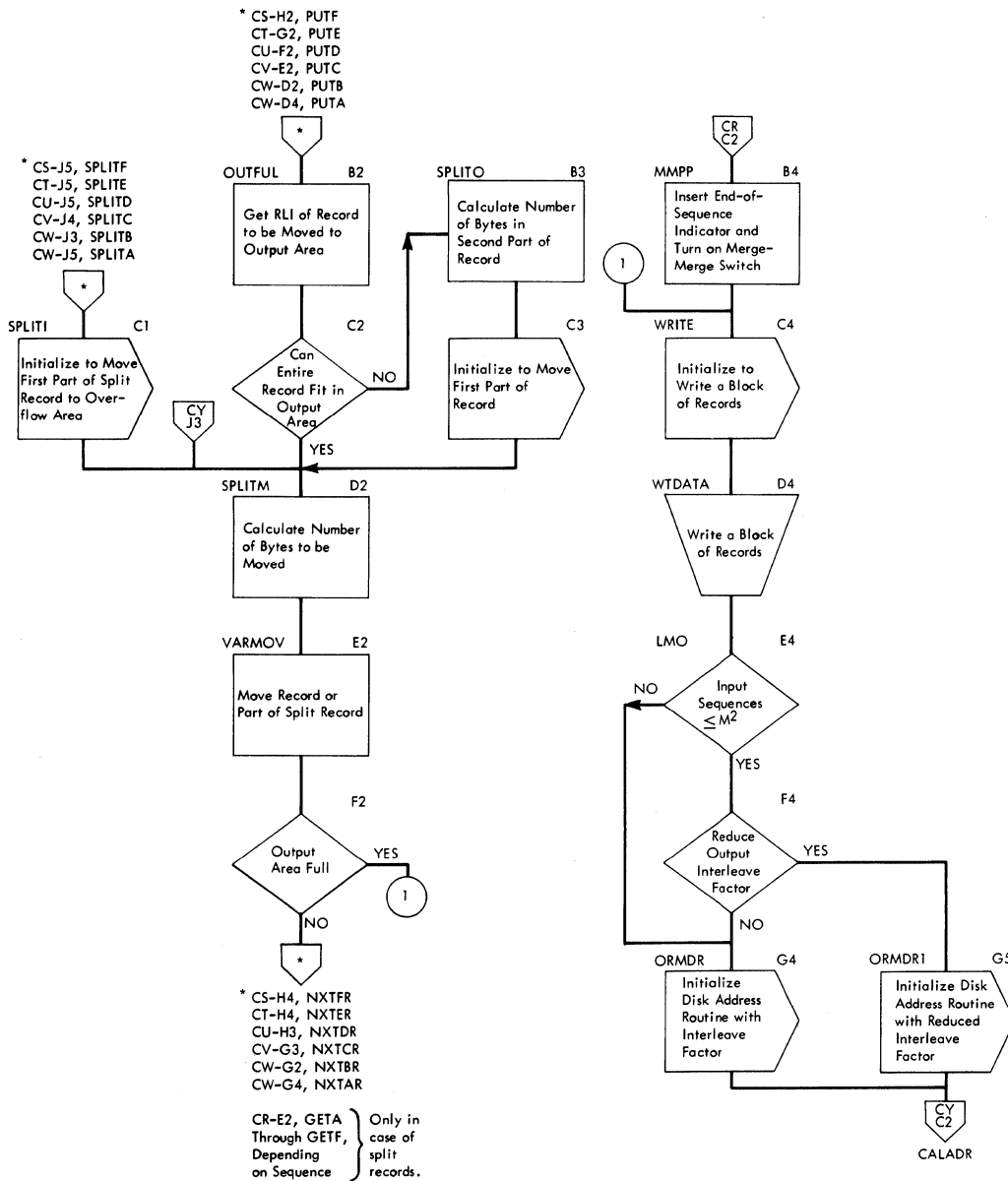


Chart CX. Output Routine, Variable-Length Records

*CR-H4, IRMDR
 CR-H5, IRMDR1
 CX-G4, ORMDR
 CX-G5, ORMDR1

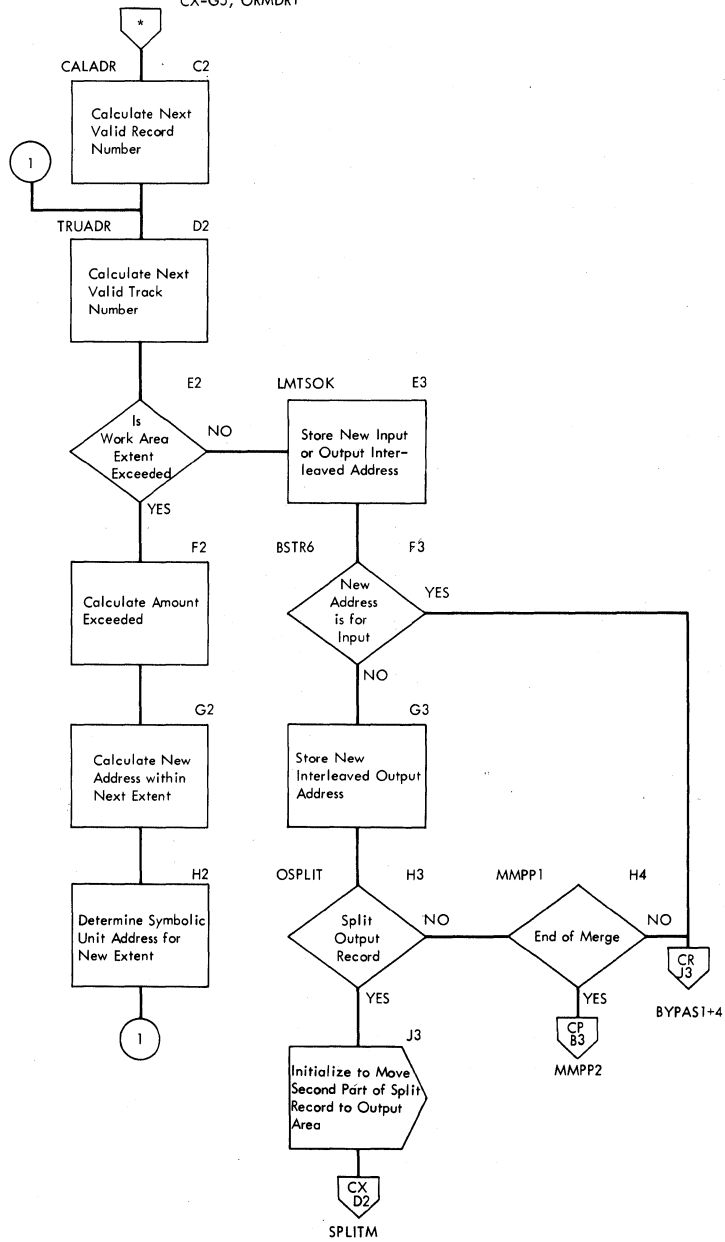


Chart CY. Calculate Interleave Disk Address, Variable-Length Records

DSORT301 - 4-WAY MERGE
 DSORT302 - 7-WAY MERGE

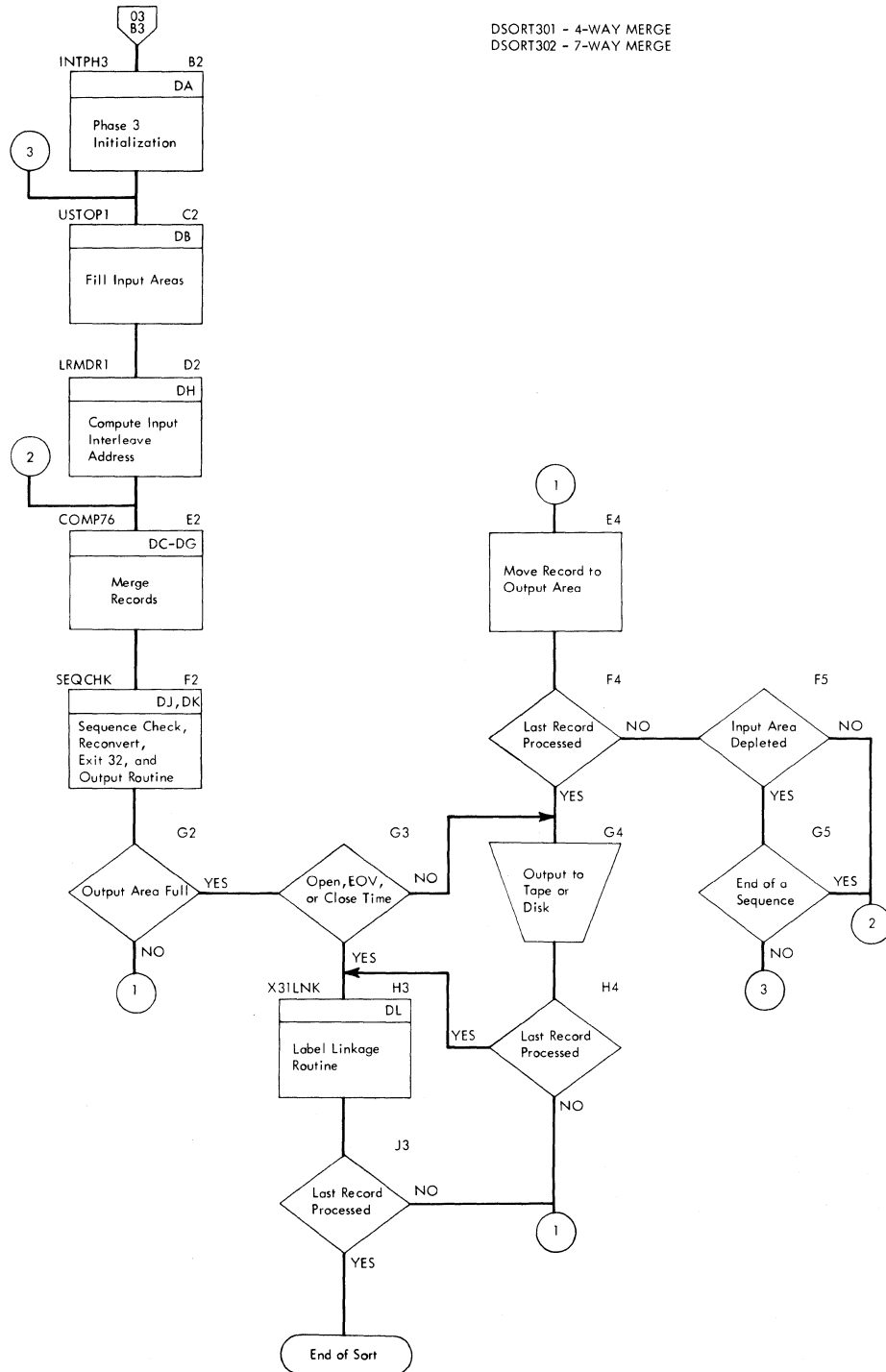


Chart 04. Final Merge, Fixed-Length Records (Phase 3), DSORT301 or DSORT302

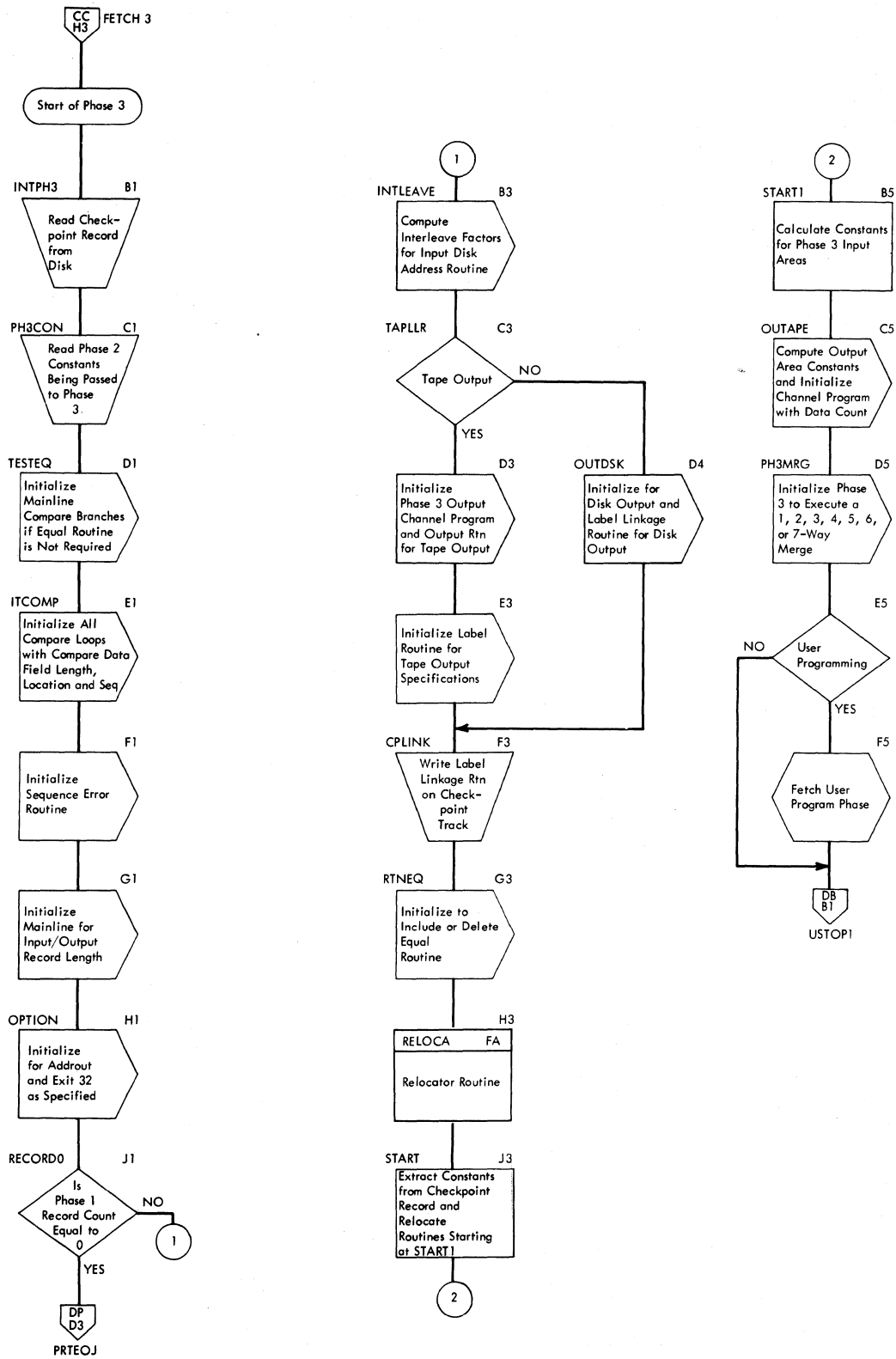


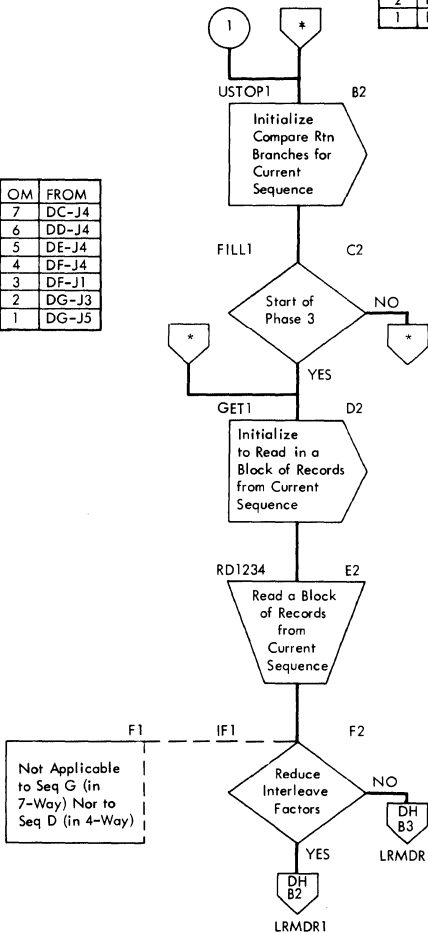
Chart DA. Phase 3 Initialization, Fixed-Length Records

OM	FROM
7	DC-J4
6	DD-J4
5	DE-J4
4	DF-J4
3	DG-J1
2	DG-J3
1	DG-J5

NOTE: USTOP1, FILL1, GET1, and IF1 Labels Apply for Input Sequence 1. For Sequences 2 Through 7, Label Suffixes are 2 Through 7 Respectively.

OM	FROM
7	DC-J4
6	DD-J4
5	DE-J4
4	DF-J4
3	DF-J1
2	DG-J3
1	DG-J5

OM	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Seq 6	Seq 7
7	DC-H2 7:1	DC-G2 7:2	DC-F2 7:3	DC-E1 7:4	DC-D2 7:5	DC-B2 7:6	DC-B2 7:6
6	DD-G2 6:1	DD-F2 6:2	DD-E2 6:3	DD-D1 6:4	DD-B2 6:5		
5	DE-F2 5:1	DE-E2 5:2	DE-D2 5:3	DE-B2 5:4			
4	DF-E2 4:1	DF-D2 4:2	DF-B2 4:3	DF-B2 4:3			
3	DG-D1 3:1	DG-B1 3:2					
2	DG-D3 2:1						



OM	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Seq 6	Seq 7
7	DC-H2 7:1	DC-G2 7:2	DC-F2 7:3	DC-E2 7:4	DC-D2 7:5	DC-B2 7:6	DC-B2 7:6
6	DD-G2 6:1	DD-F2 6:2	DD-E2 6:3	DD-D2 6:4	DD-B2 6:5	DD-B2 6:5	
5	DE-F2 5:1	DE-E2 5:2	DE-D2 5:3	DE-B2 5:4	DE-B2 5:4		
4	DF-E2 4:1	DF-D2 4:2	DF-B2 4:3	DF-B2 4:3			
3	DG-D1 3:1	DG-B1 3:2	DG-B1 3:2				
2	DG-C3 2:1	DG-C3 2:1					
1	DG-E5 PUT1						
Init	DB-B2	DB-B2	DB-B2	DB-B2	DB-B2	DB-B2	DC-B2
Read-in	USTOP2	USTOP3	USTOP4	USTOP5	USTOP 6	USTOP7	COMP76

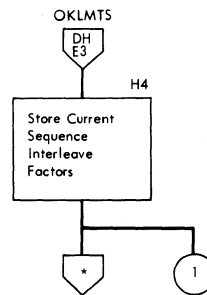


Chart DB. Input Routine, Fixed-Length Records

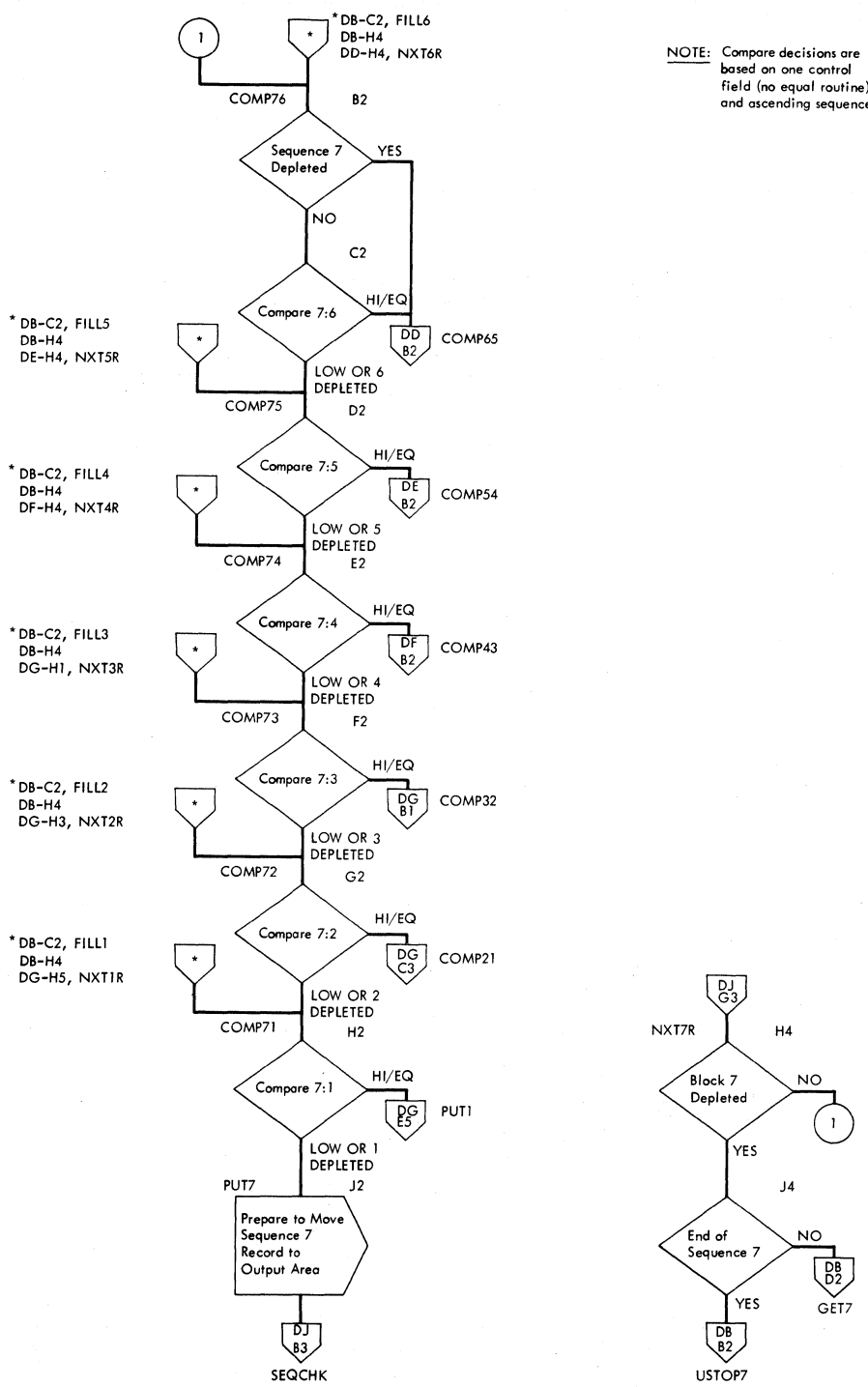
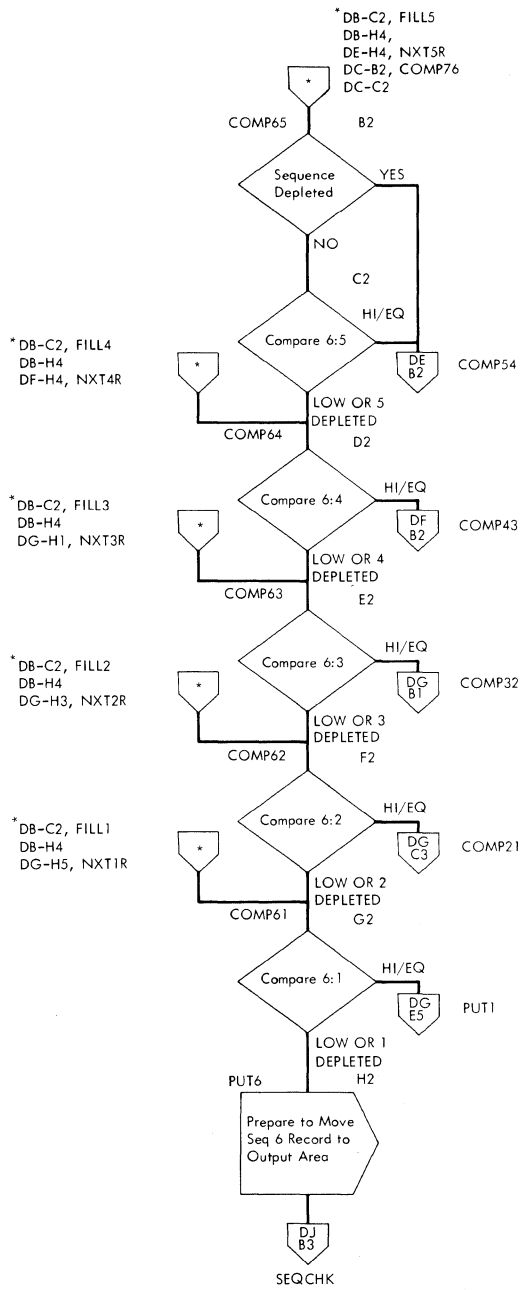


Chart DC. Mainline Compare Routine, Fixed-Length Records



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

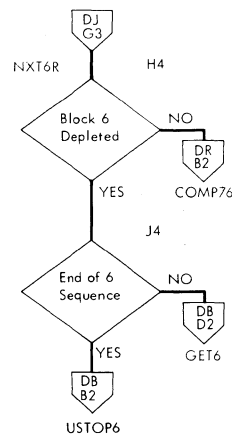
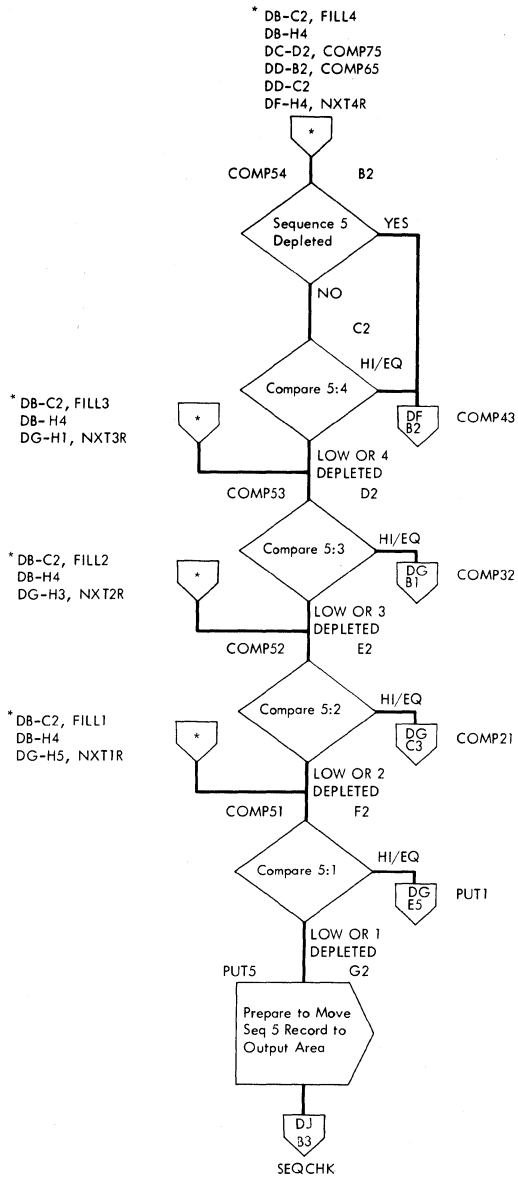


Chart DD. Mainline Compare Routine Fixed-Length Records (Cont'd)



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

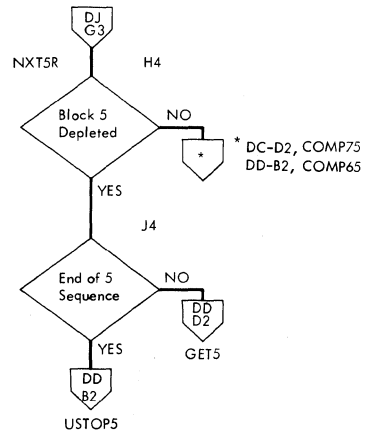
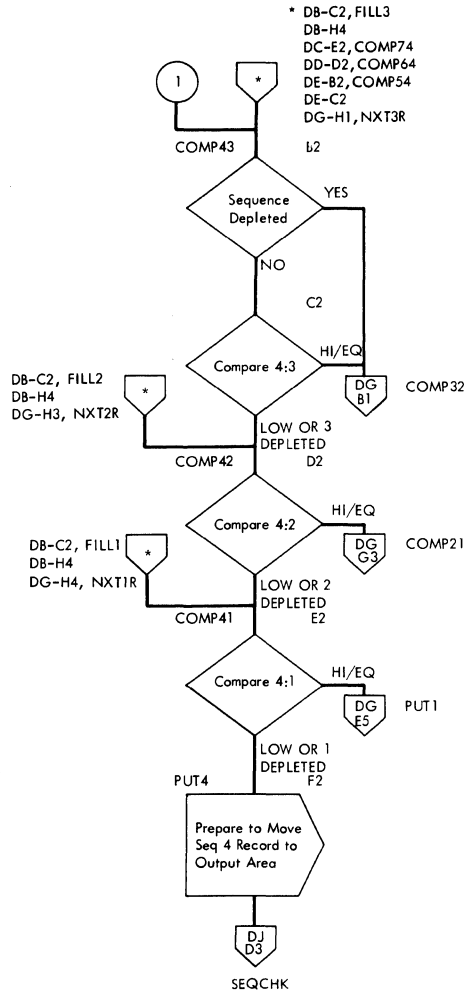


Chart DE. Mainline Compare Routine Fixed-Length Records (Cont'd)



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

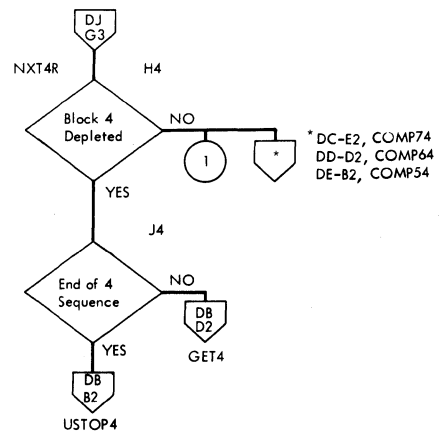


Chart DF. Mainline Compare Routine Fixed-Length Records (Cont'd)

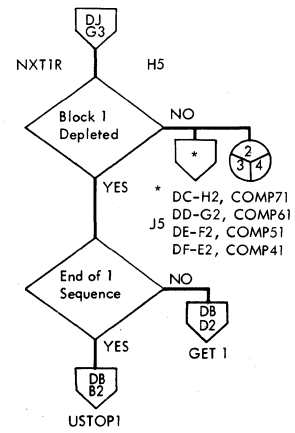
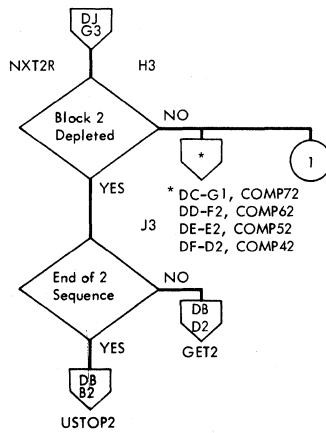
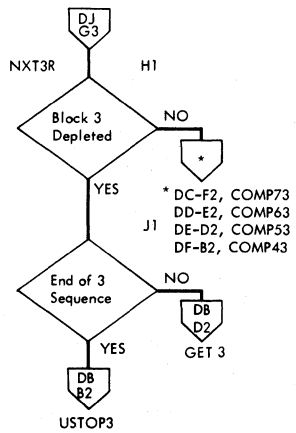
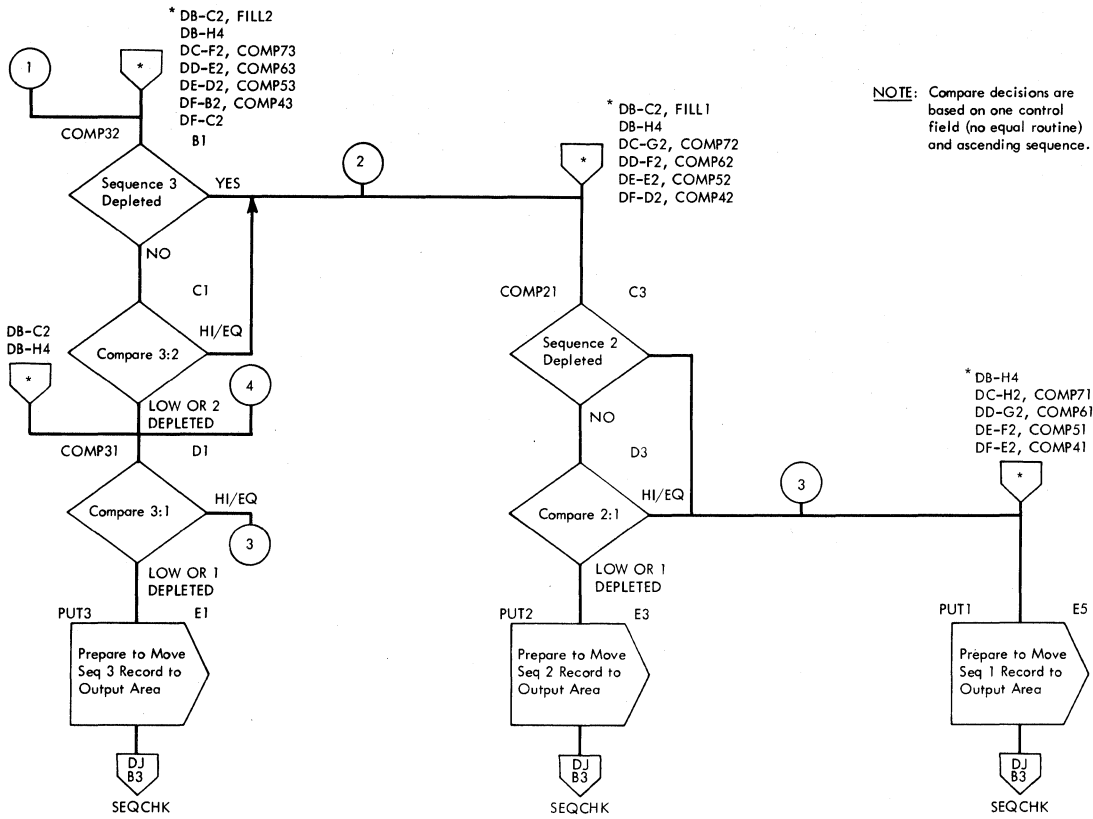


Chart DG. Mainline Compare Routine Fixed-Length Records (Cont'd)

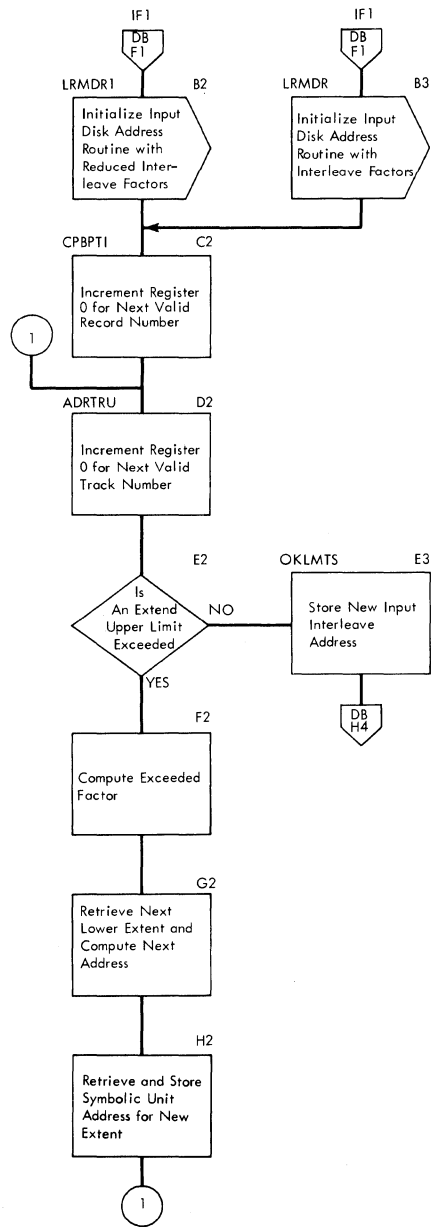


Chart DH. Compute Input Interleaved Disk Address, Fixed-Length Records

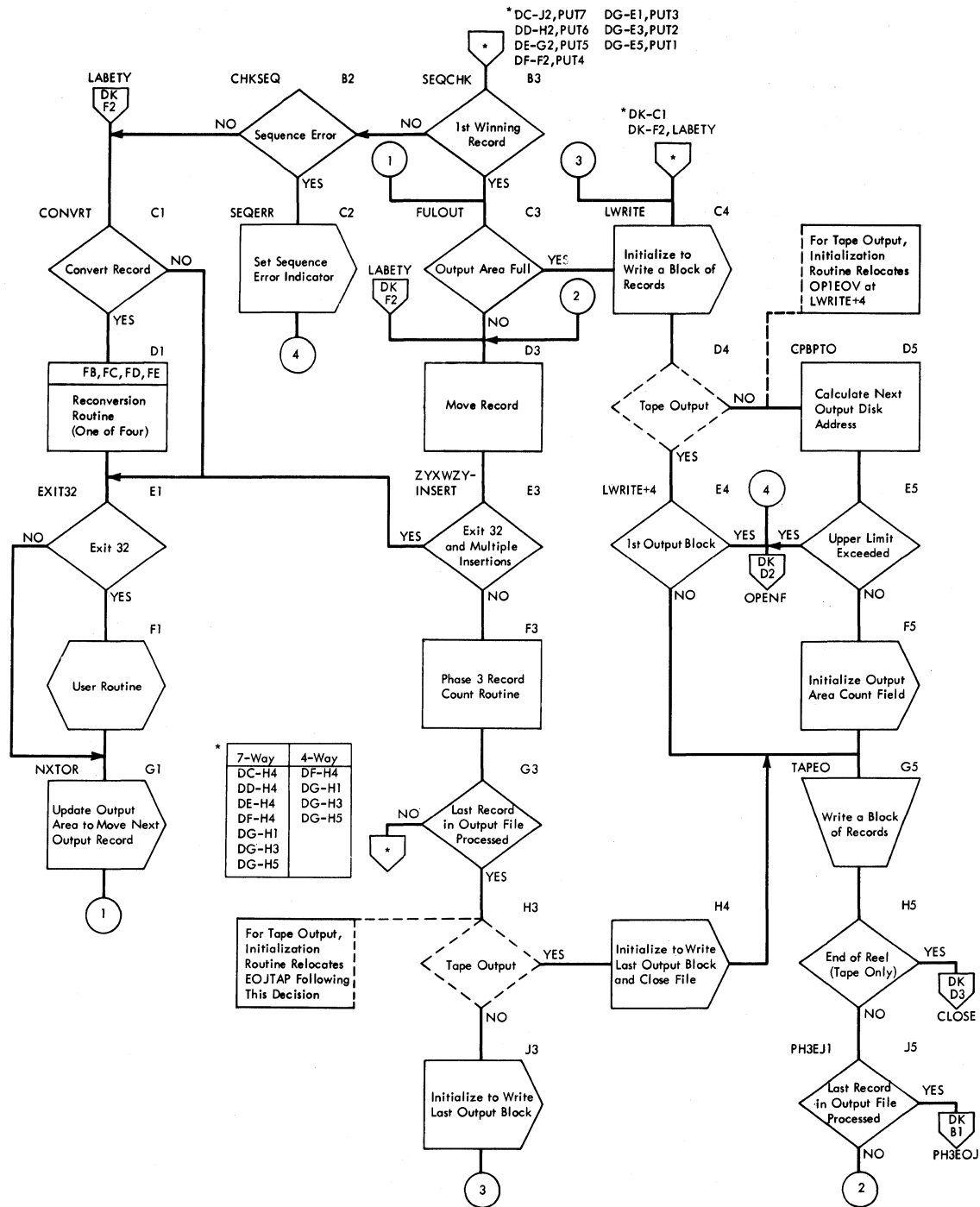


Chart DJ. Output Routine, Fixed-Length Records

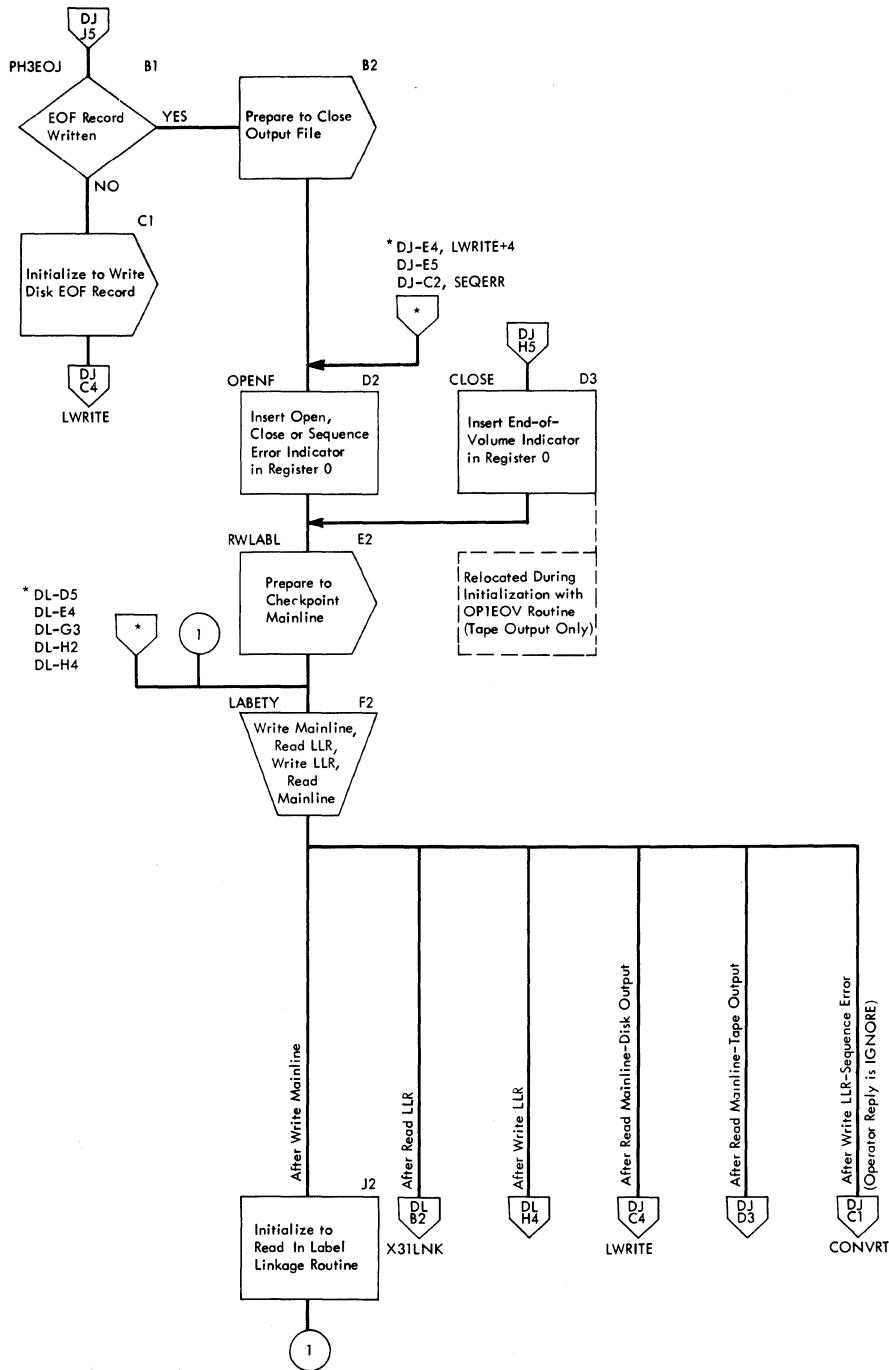


Chart DK. Output Routine, Fixed-Length Records (Cont'd)

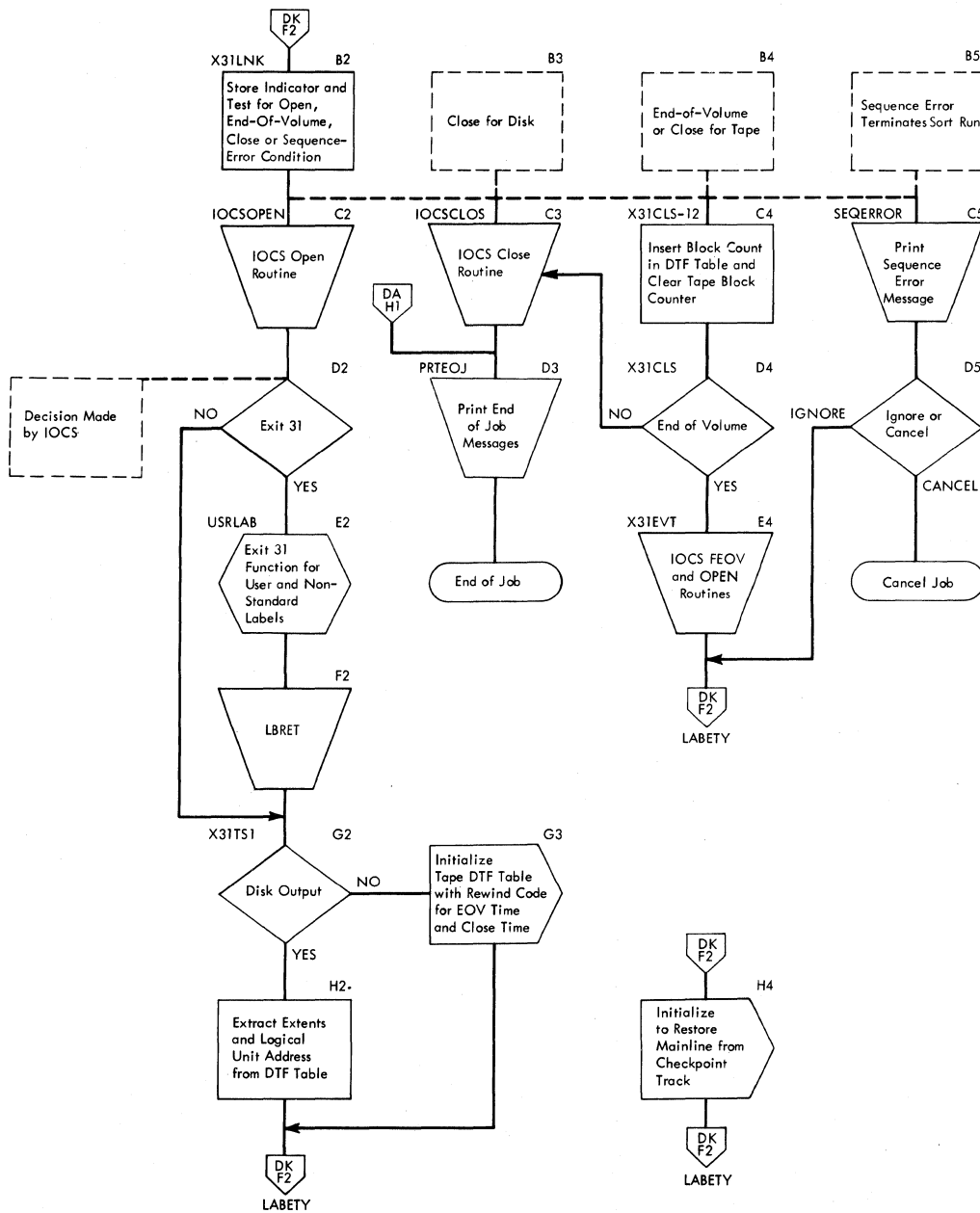


Chart DL. Label Linkage Routine (LLR), Fixed-Length Records

DSORT303 - 3-WAY MERGE
 DSORT304 - 6-WAY MERGE

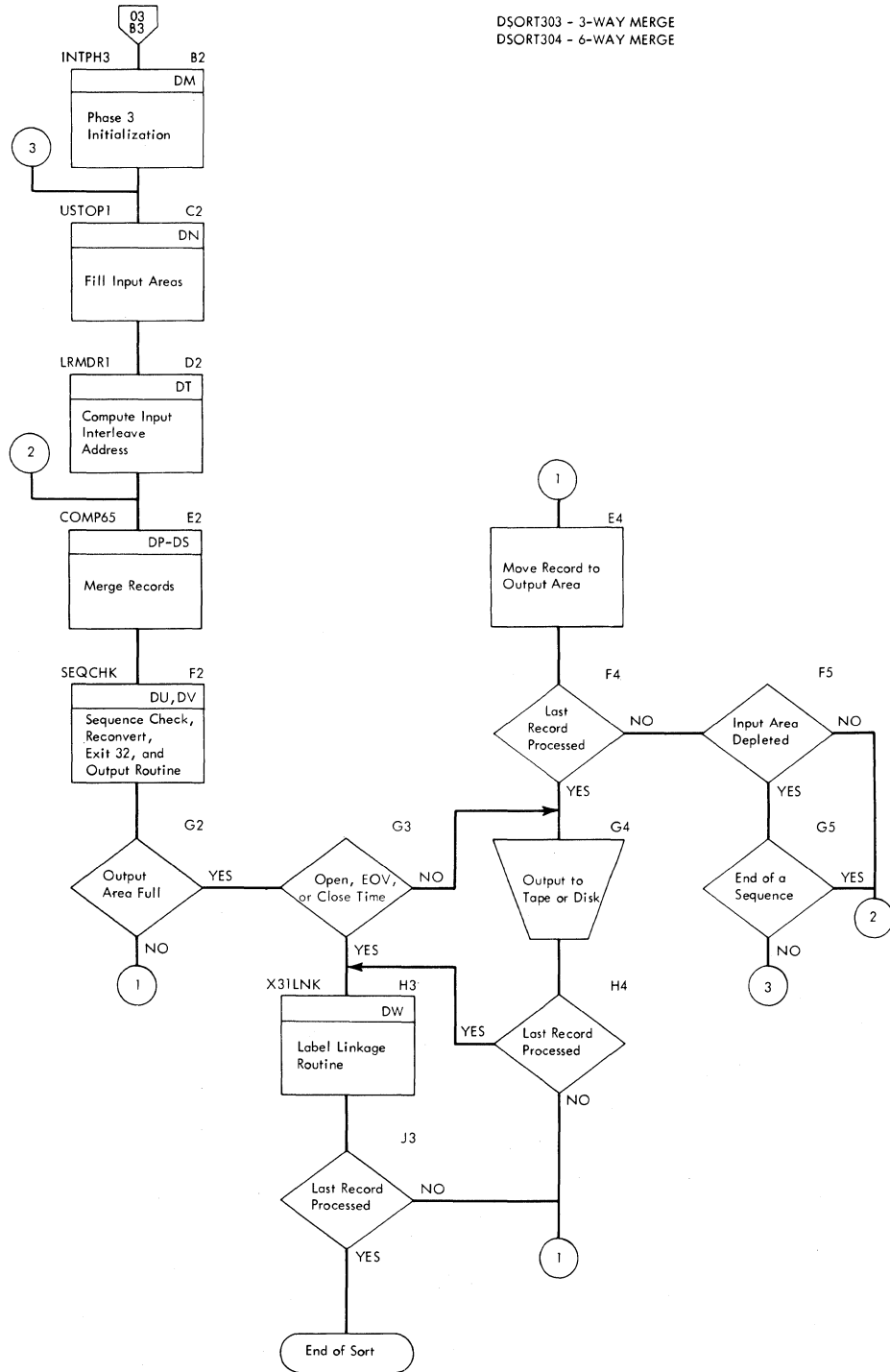


Chart 04. Final Merge, Variable-Length Records (Phase 3), DSORT303 or DSORT304

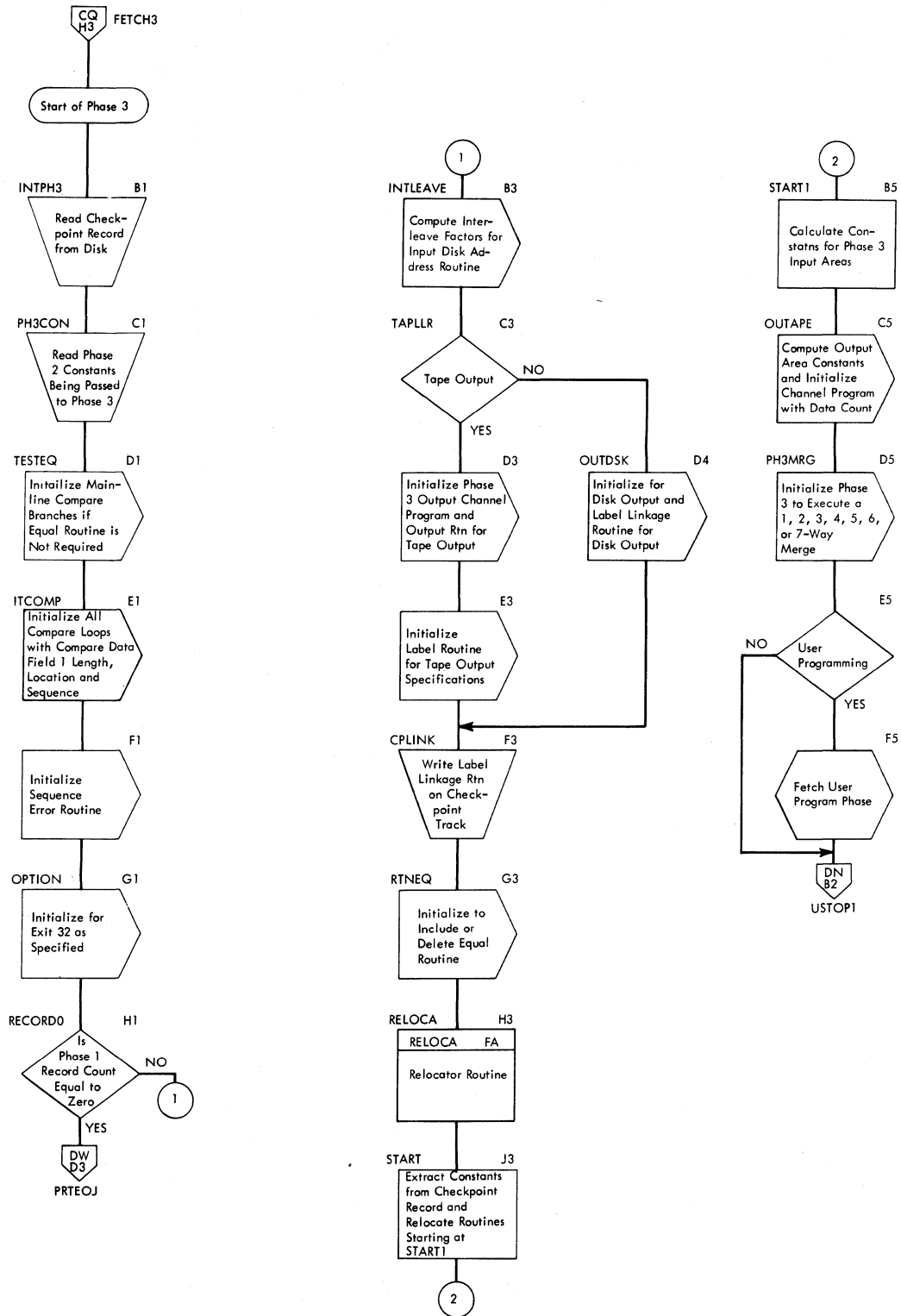
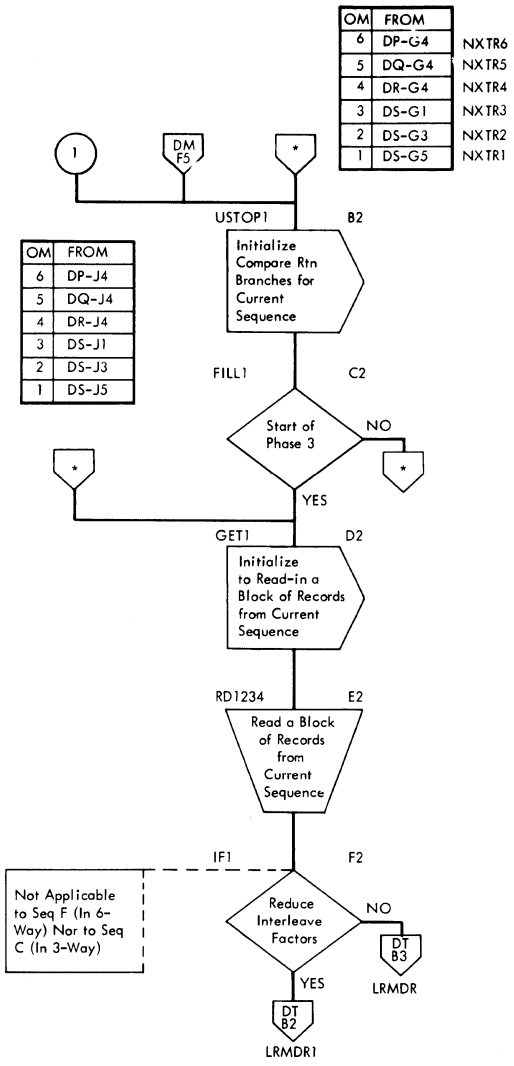


Chart DM. Phase 3 Initialization, Variable-Length Records



NOTE: USTOP1, FILL1, GET1, and IF1 labels apply for input sequence 1. For input sequences 2 through 6, label suffixes are 2 through 6 respectively.

OM	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Seq 6
6	DP-G2 6:1	DP-F2 6:2	DP-E2 6:3	DP-D2 6:4	DP-B2 6:5	DP-B2 6:5
5	DQ-F2 5:1	DQ-E2 5:2	DQ-D2 5:3	DQ-B2 5:4		
4	DR-E2 4:1	DR-D2 4:2	DR-B2 4:3			
3	DS-C1 3:1	DS-A1 3:2				
2	DS-B3 2:1					

OM	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Seq 6
6	DP-G2 6:1	DP-F2 6:2	DP-E2 6:3	DP-D2 6:4	DP-B2 6:5	DP-B2 6:5
5	DQ-F2 5:1	DQ-E2 5:2	DQ-D2 5:3	DQ-B2 5:4	DQ-B2 5:4	
4	DR-E2 4:1	DR-D2 4:2	DR-B2 4:3	DR-B2 4:3		
3	DS-C1 3:1	DS-A1 3:2	DS-A1 3:2			
2	DS-B3 2:1	DS-B3 2:1				
1	DS-D5PUT1					

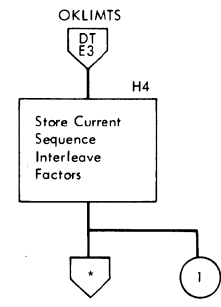
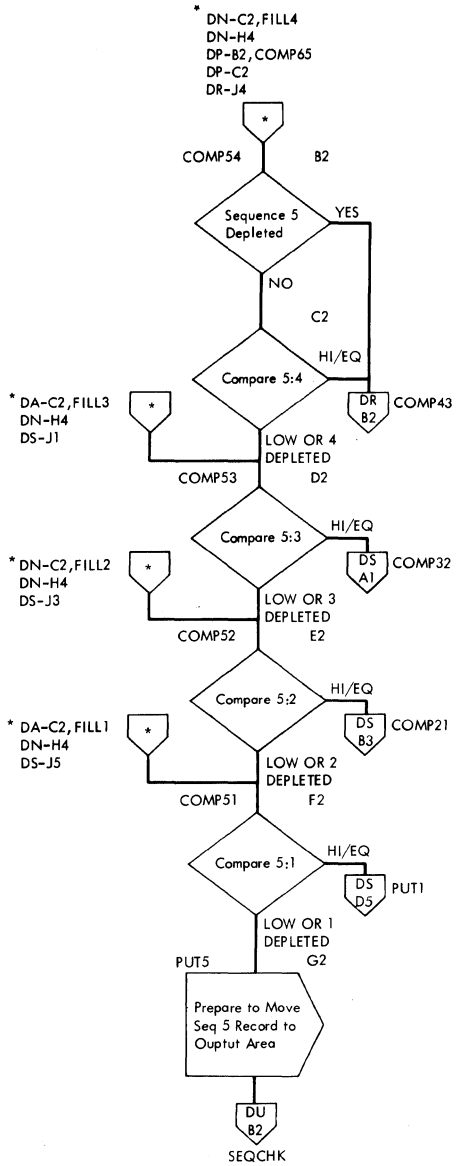


Chart DN. Input Routine, Variable-Length Records



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

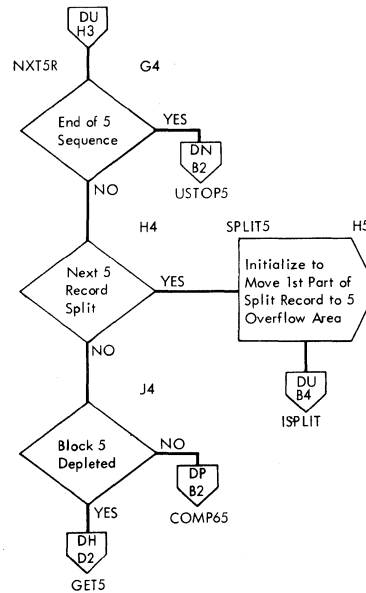
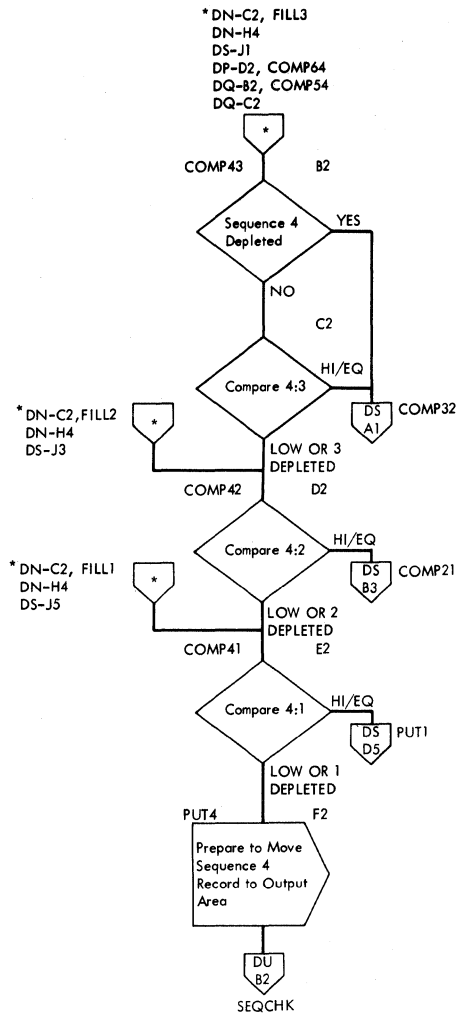


Chart DQ. Mainline Compare Routine, Variable-Length Records (Cont'd)



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

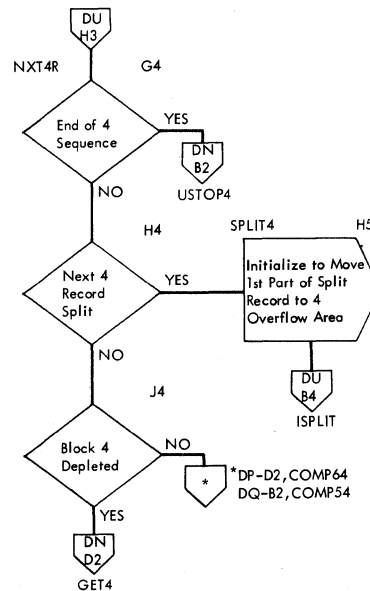


Chart DR. Mainline Compare Routine, Variable-Length Records (Cont'd)

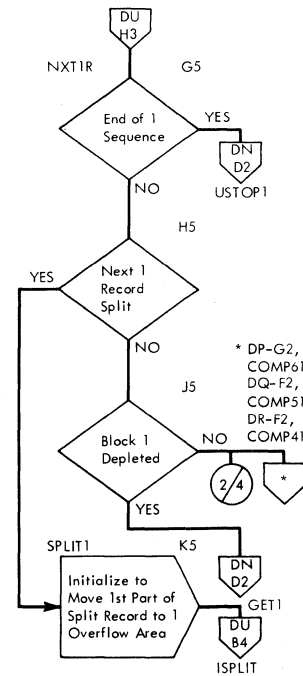
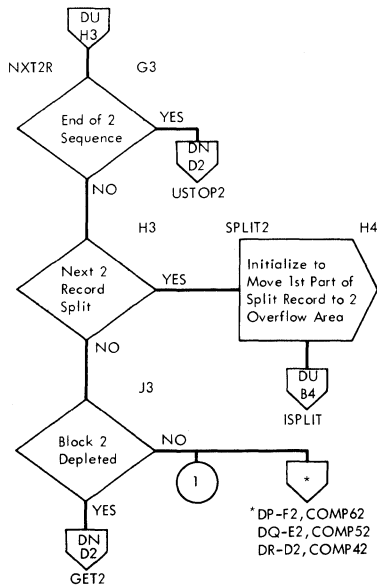
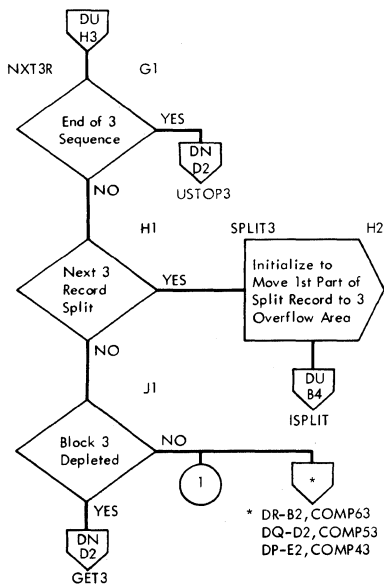
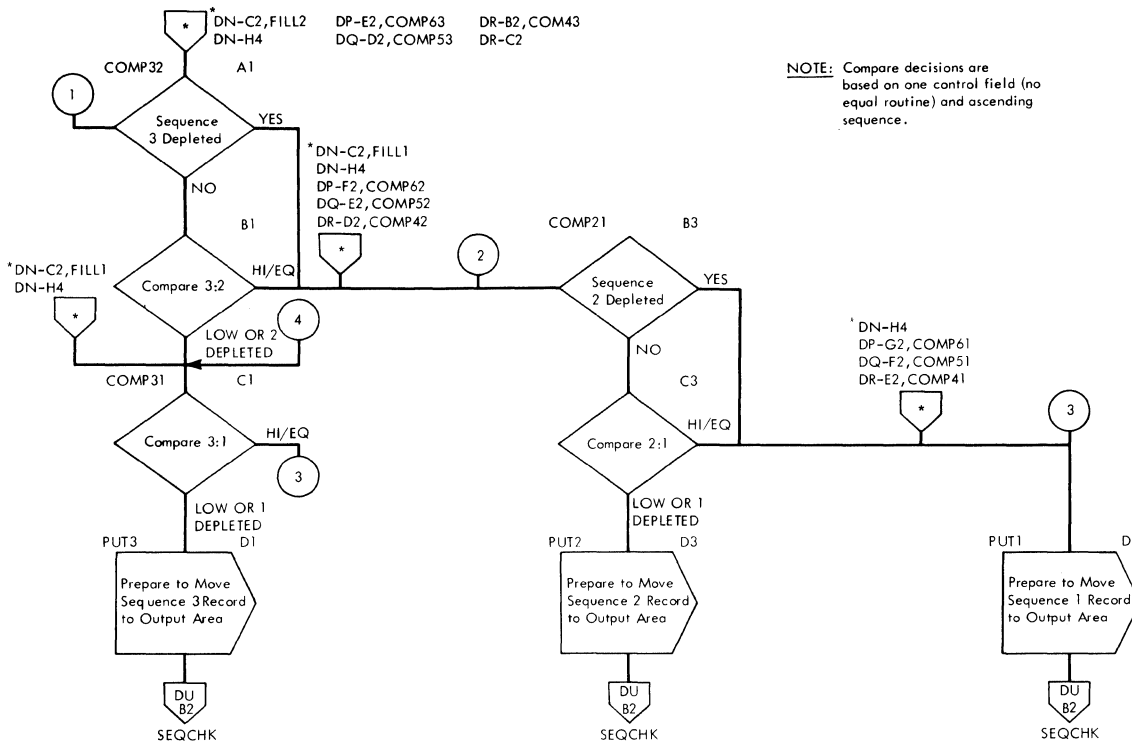


Chart DS. Mainline Compare Routine, Variable-Length Records (Cont'd)

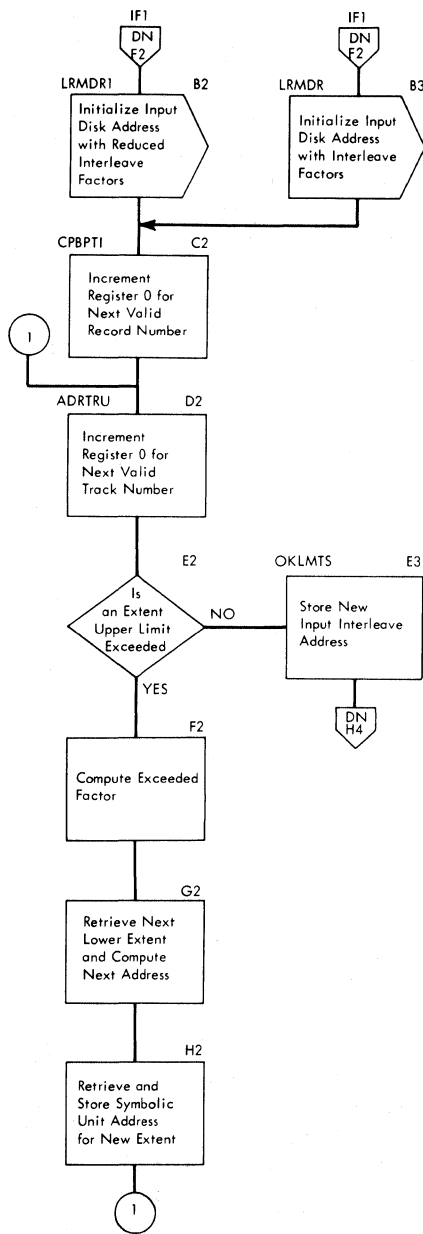


Chart DT. Compute Input Interleaved Disk Address, Variable-Length Records

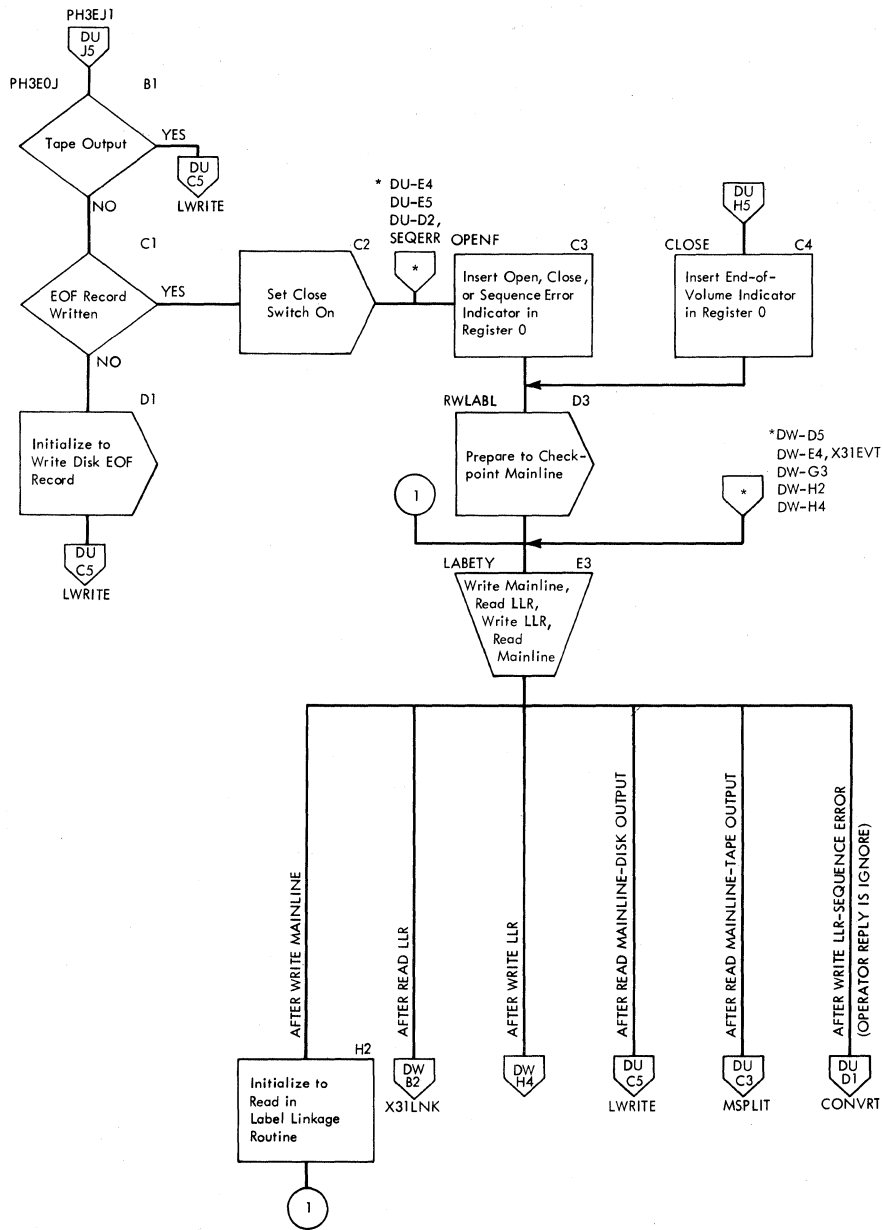


Chart DV. Output Routine, Variable-Length Records (Cont'd)

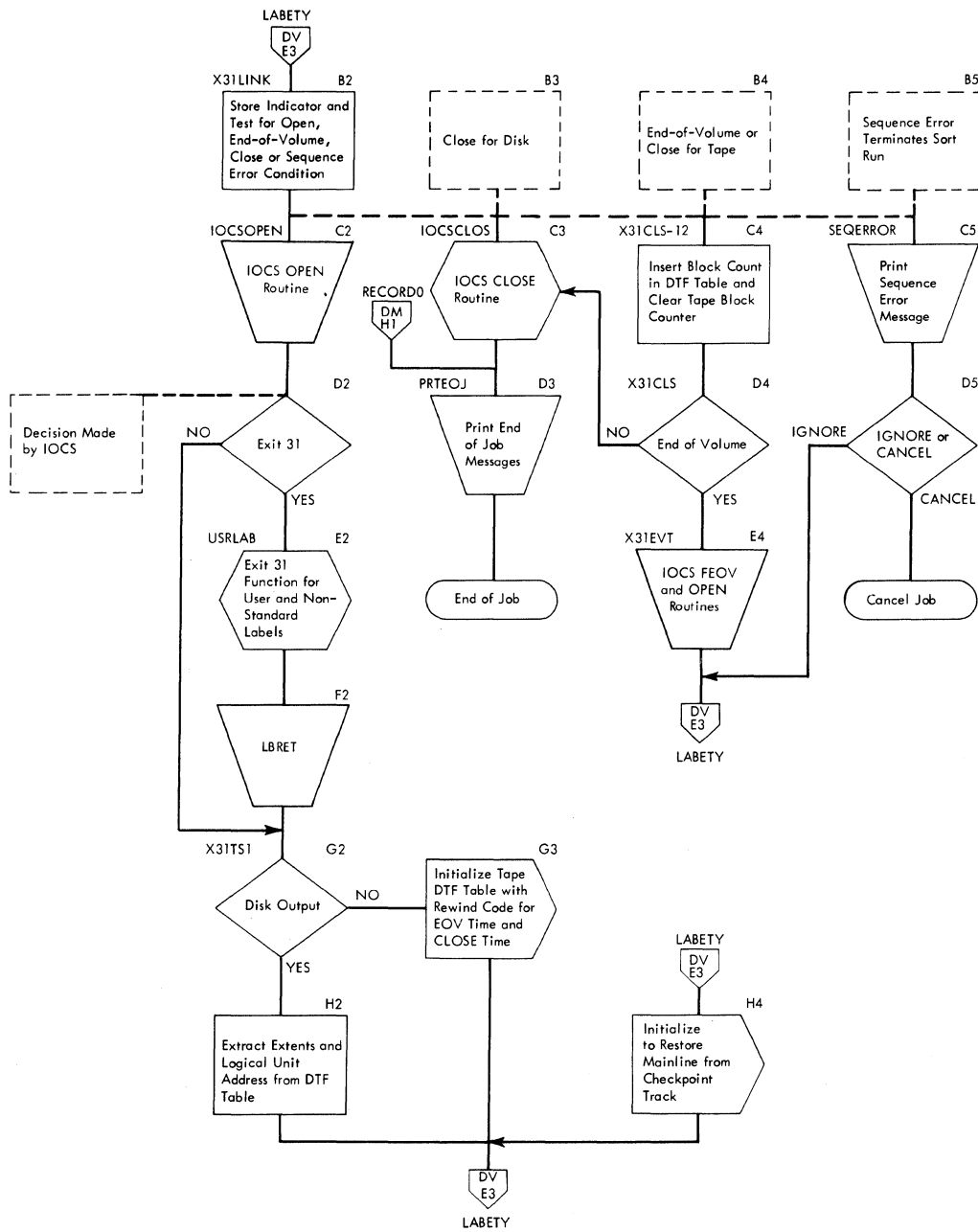
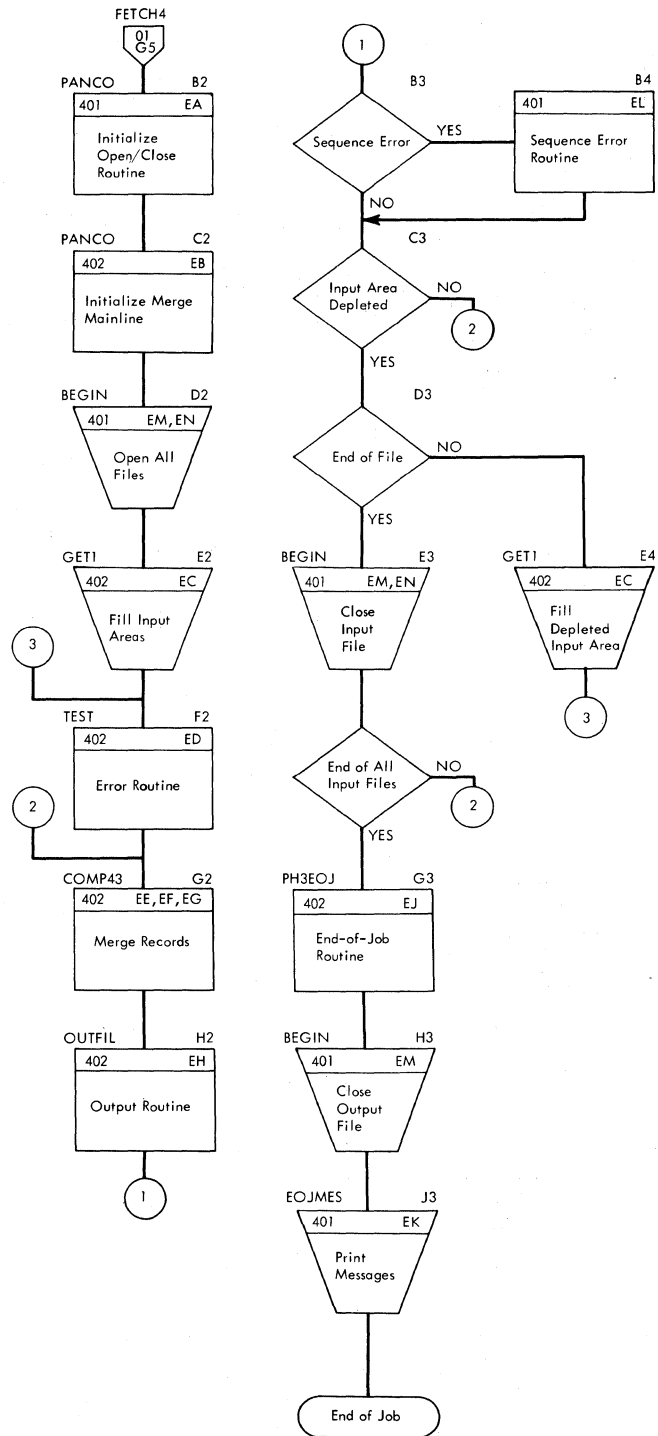


Chart DW. Label Linkage Routine (LLR), Variable-Length Records



NOTE: 401 and 402 in blocks denote DSORTnnn in which routine appears. Linkage between the overlays is provided by checkpoint routine (Chart EP).

Chart 05. Merge-Only (Phase 4), DSORT401 and DSORT402

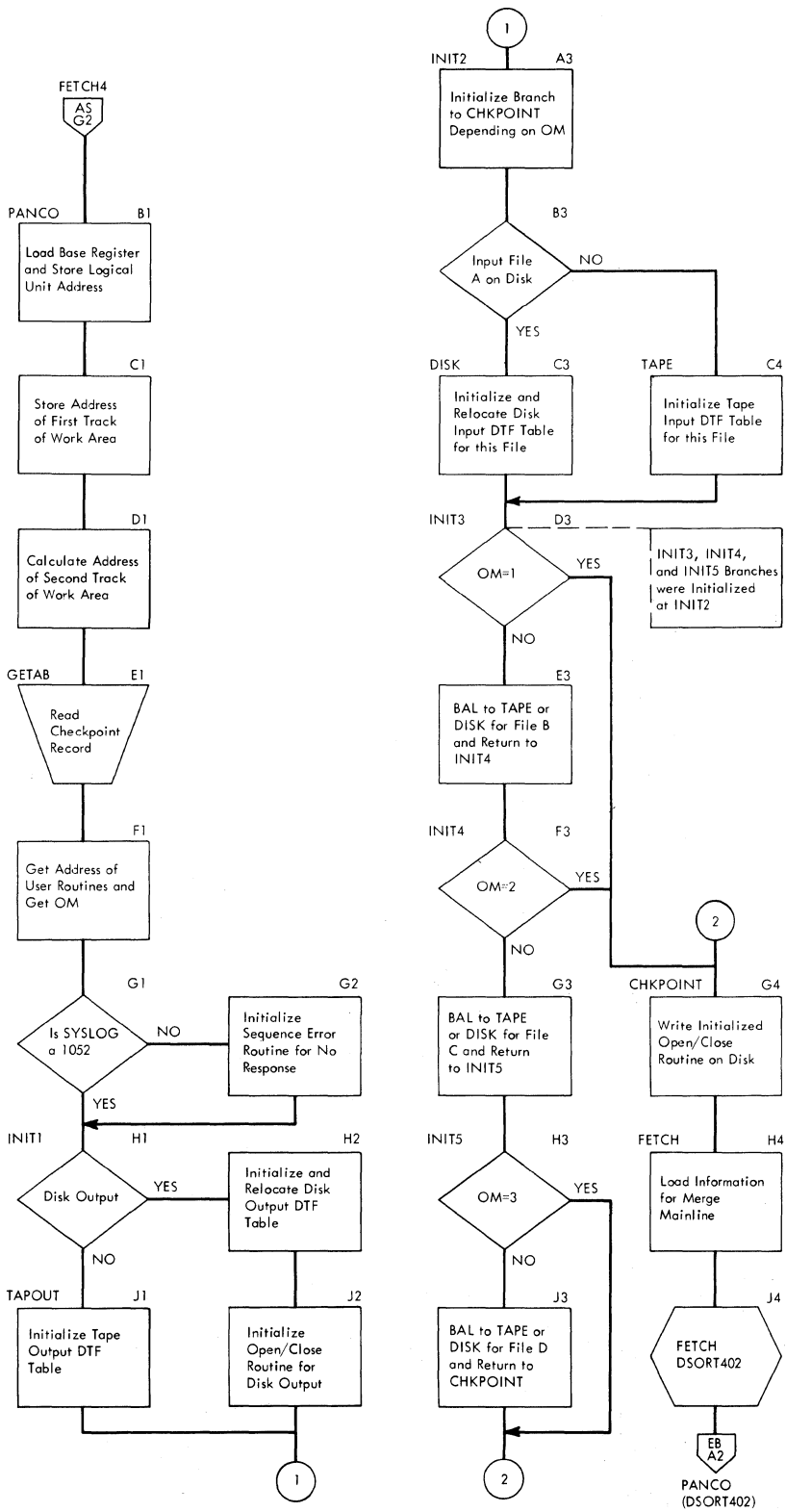


Chart EA. Initialize Open/Close Routine, DSORT401

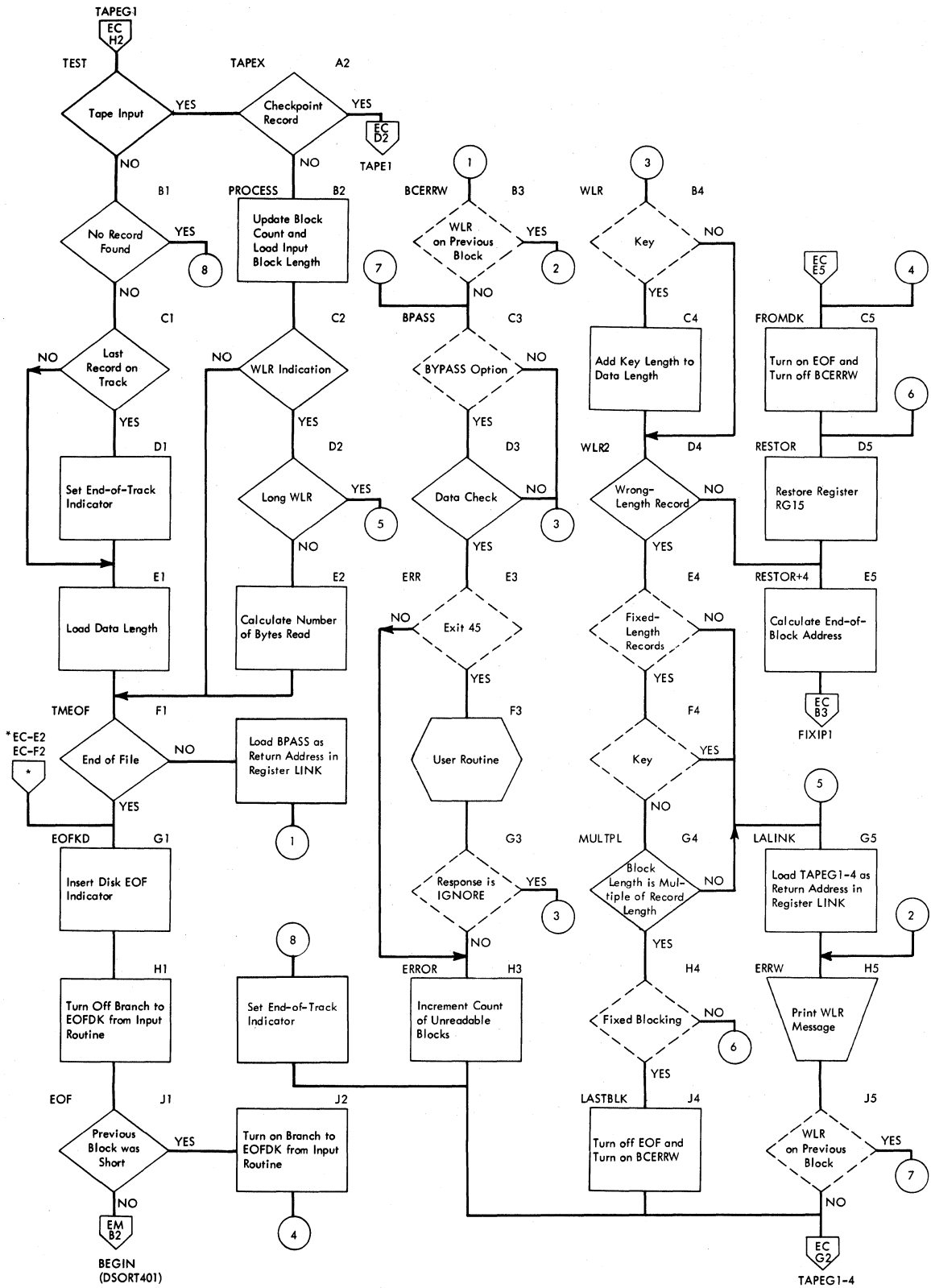
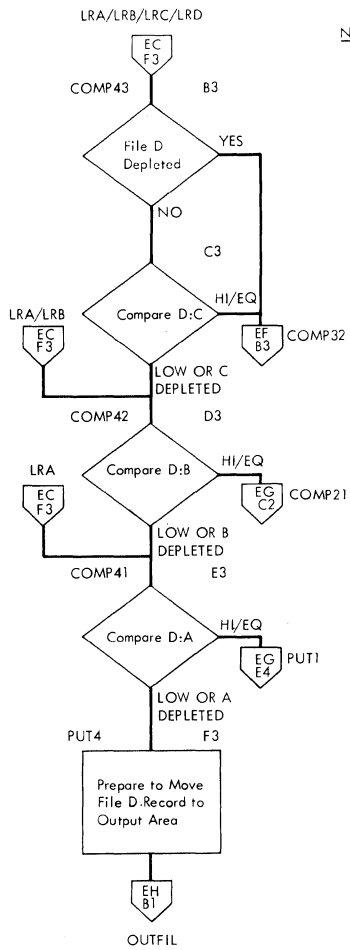


Chart ED. Error Routine, DSORT402



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

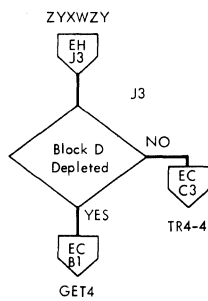
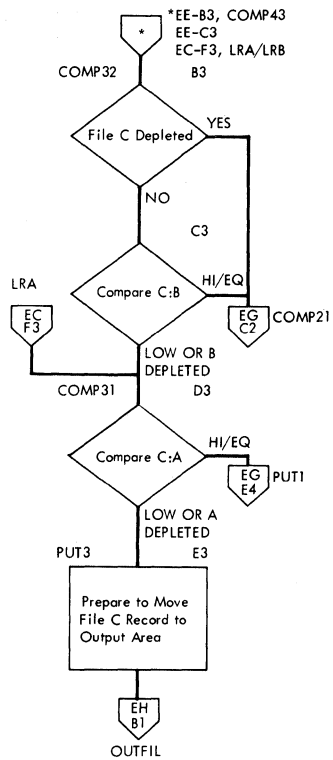


Chart EE. File D Compare Loop, DSORT402



NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

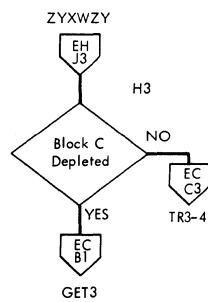


Chart EF. File C Compare Loop, DSORT402

NOTE: Compare decisions are based on one control field (no equal routine) and ascending sequence.

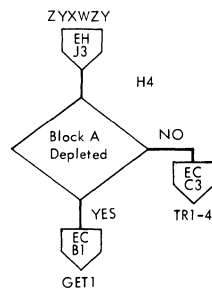
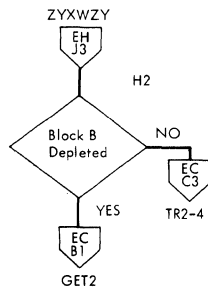
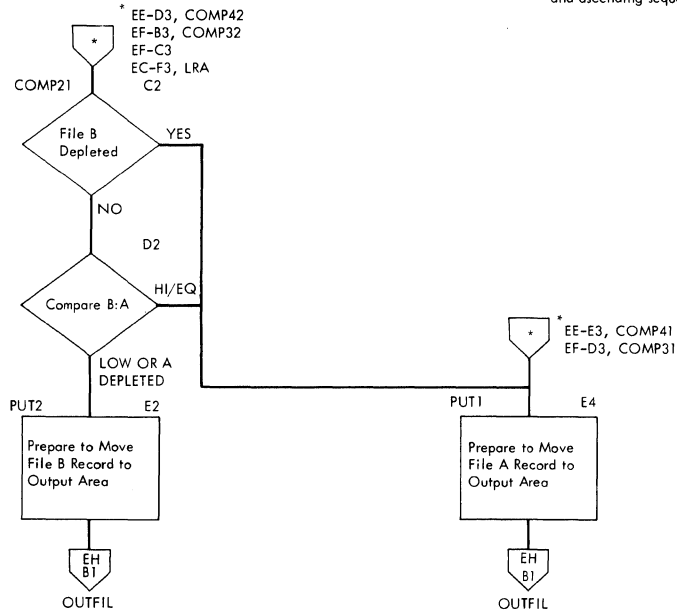


Chart EG. File B Compare Loop, DSORT402

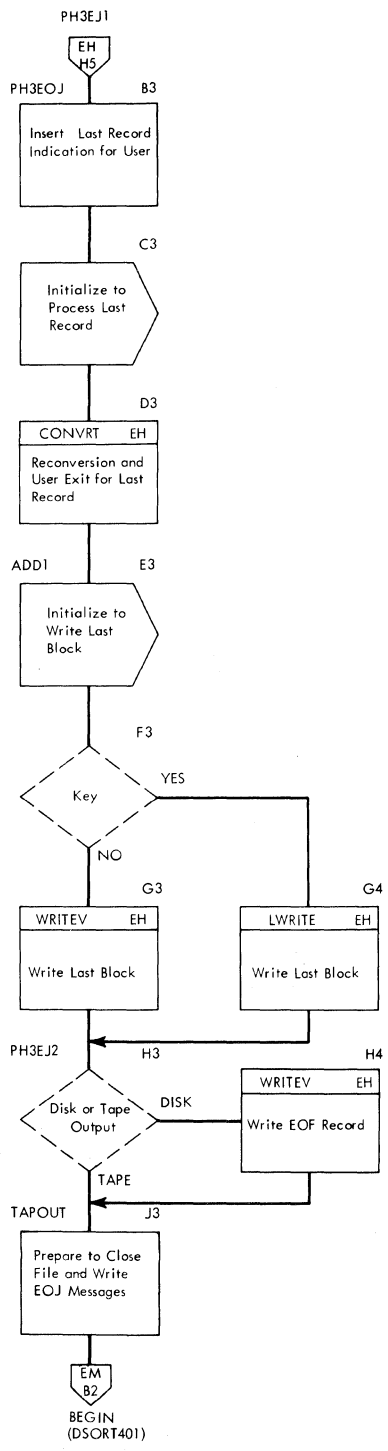


Chart EJ. End of Job Routine, DSORT402

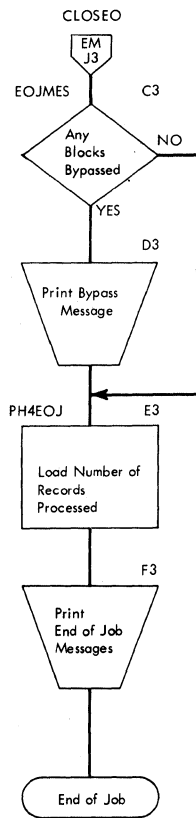


Chart EK. End of Job Messages, DSORT401

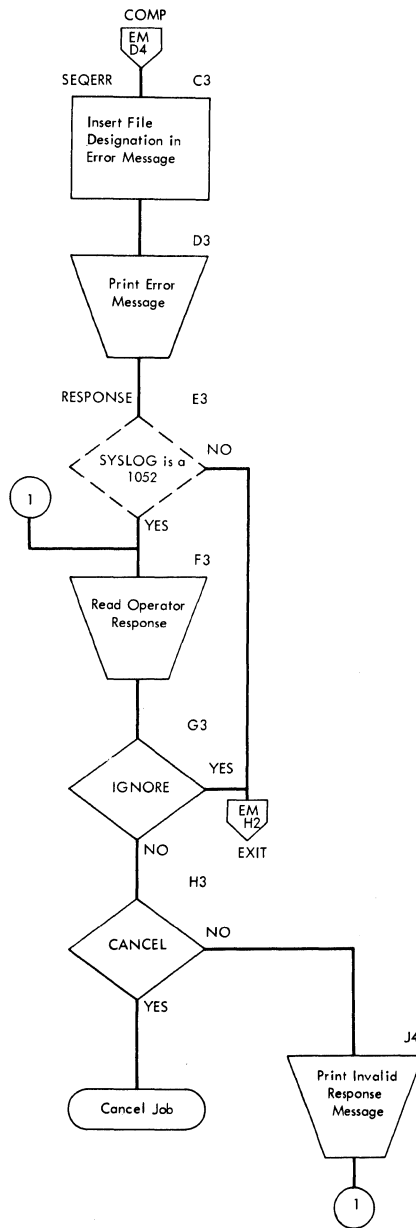


Chart EL. Sequence Error Routine, DSORT401

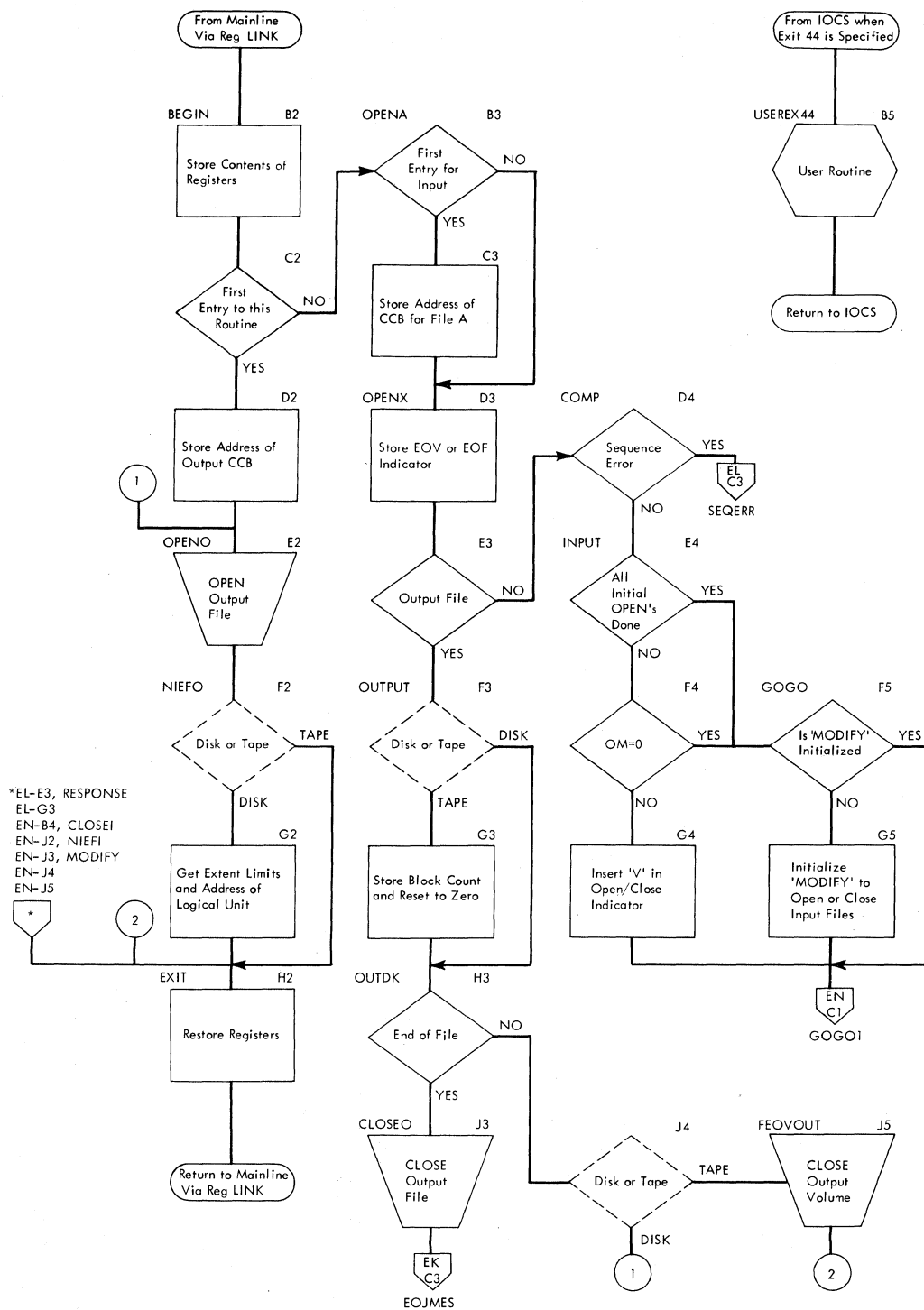


Chart EM. Open/Close Routine, DSORT401

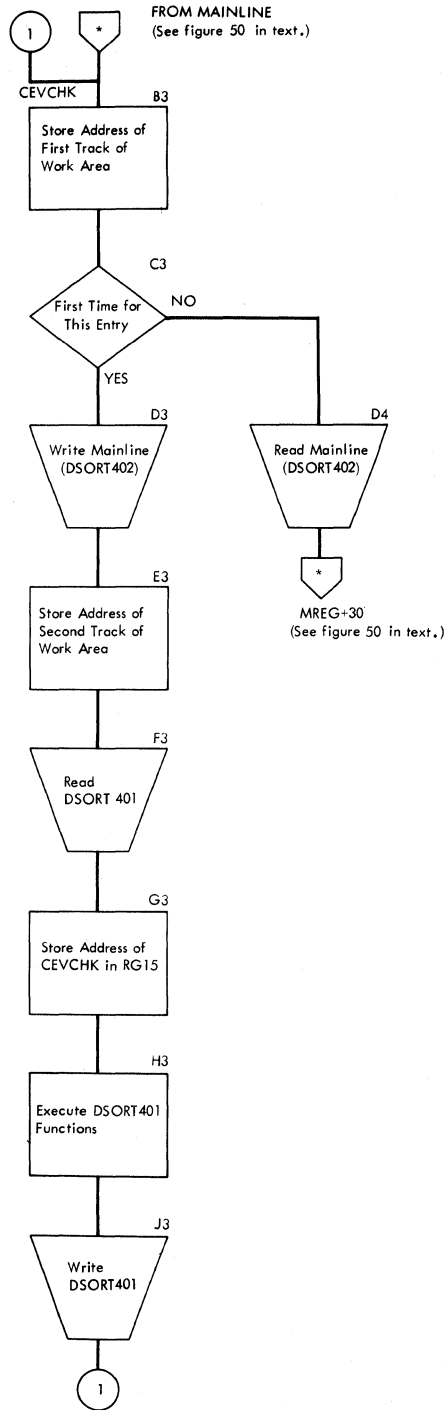


Chart EP. Checkpoint Routine, DSORT402

ENTER FROM MAINLINE
INITIALIZATION VIA LINK REGISTER

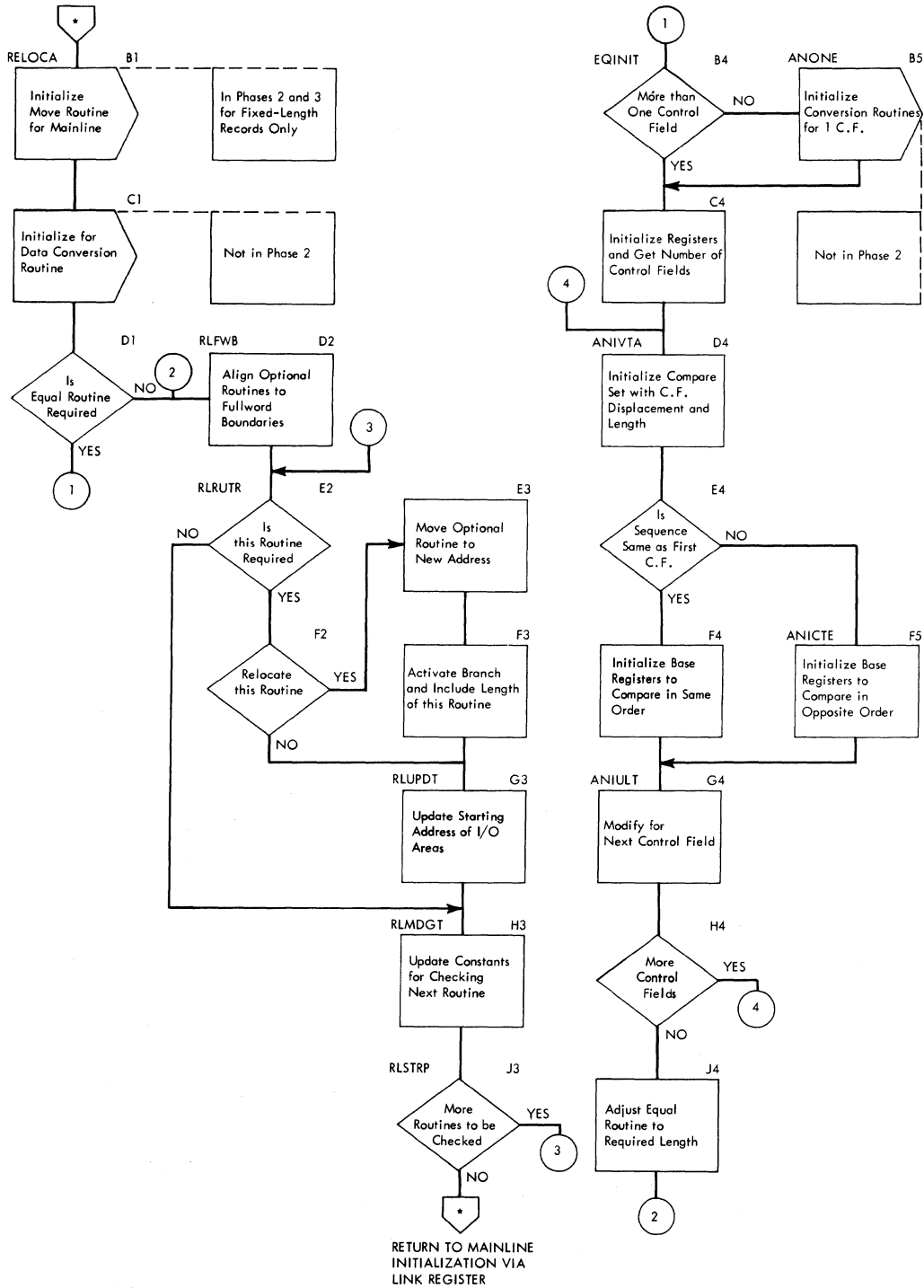


Chart FA. Relocator Routine, Phases 2, 3, and 4

BAL FROM MAINLINE
 (PHASES 1 AND 3 FOR SORT,
 PHASE 4 FOR MERGE-ONLY)

NOTE: Convert in
 phases 1 and 4.
 Reconvert in
 phases 3 and 4.

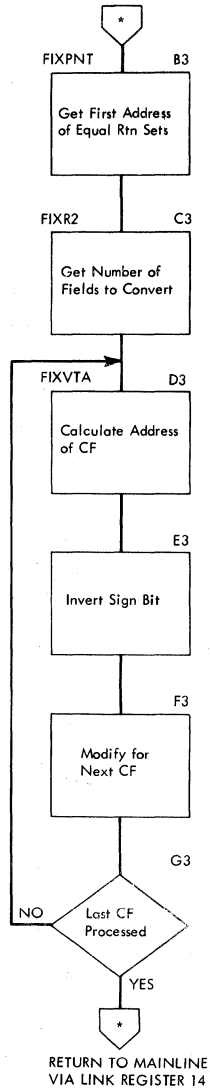


Chart FB. Fixed Point Convert/Reconvert

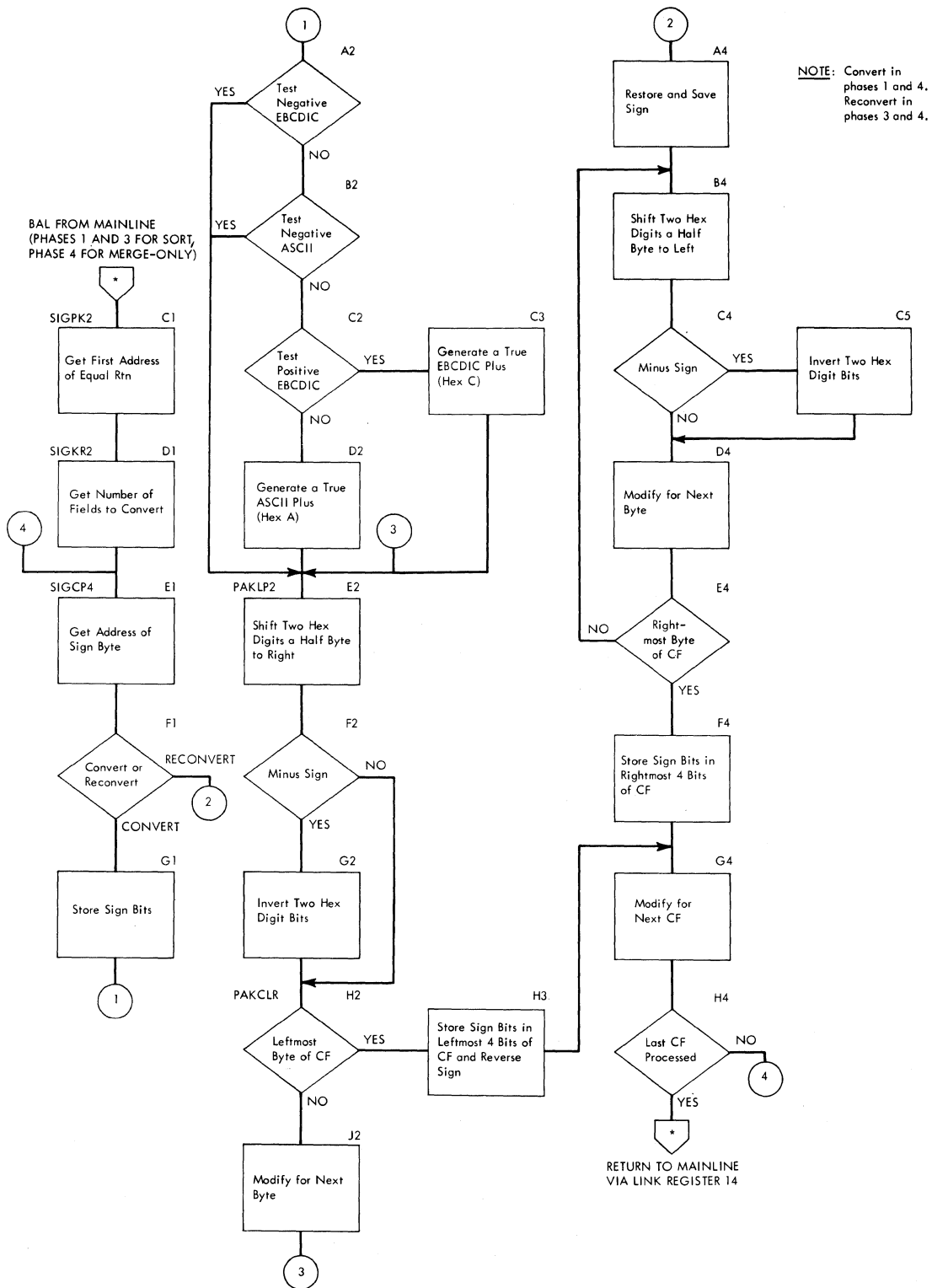


Chart FC. Floating-Point Convert/Reconvert

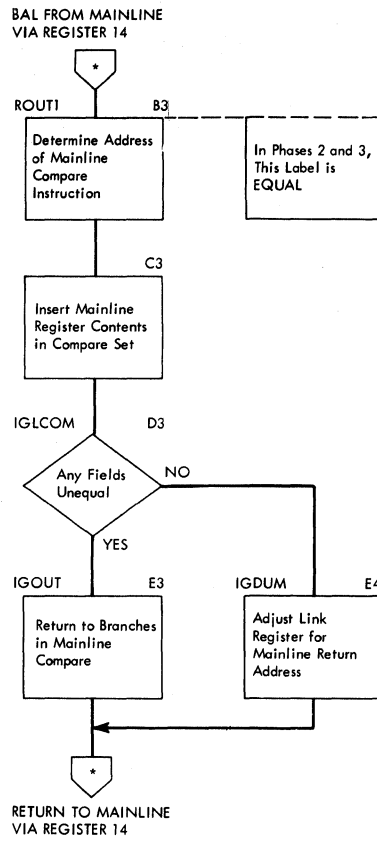


Chart FD. Packed-Decimal (SIGPAK) Convert/Reconvert

BAL FROM MAINLINE
 (PHASE 1 AND 3 FOR SORT
 PHASE 4 FOR MERGE-ONLY)

NOTE: Convert in
 phases 1 and 4.
 Reconvert in
 phases 3 and 4.

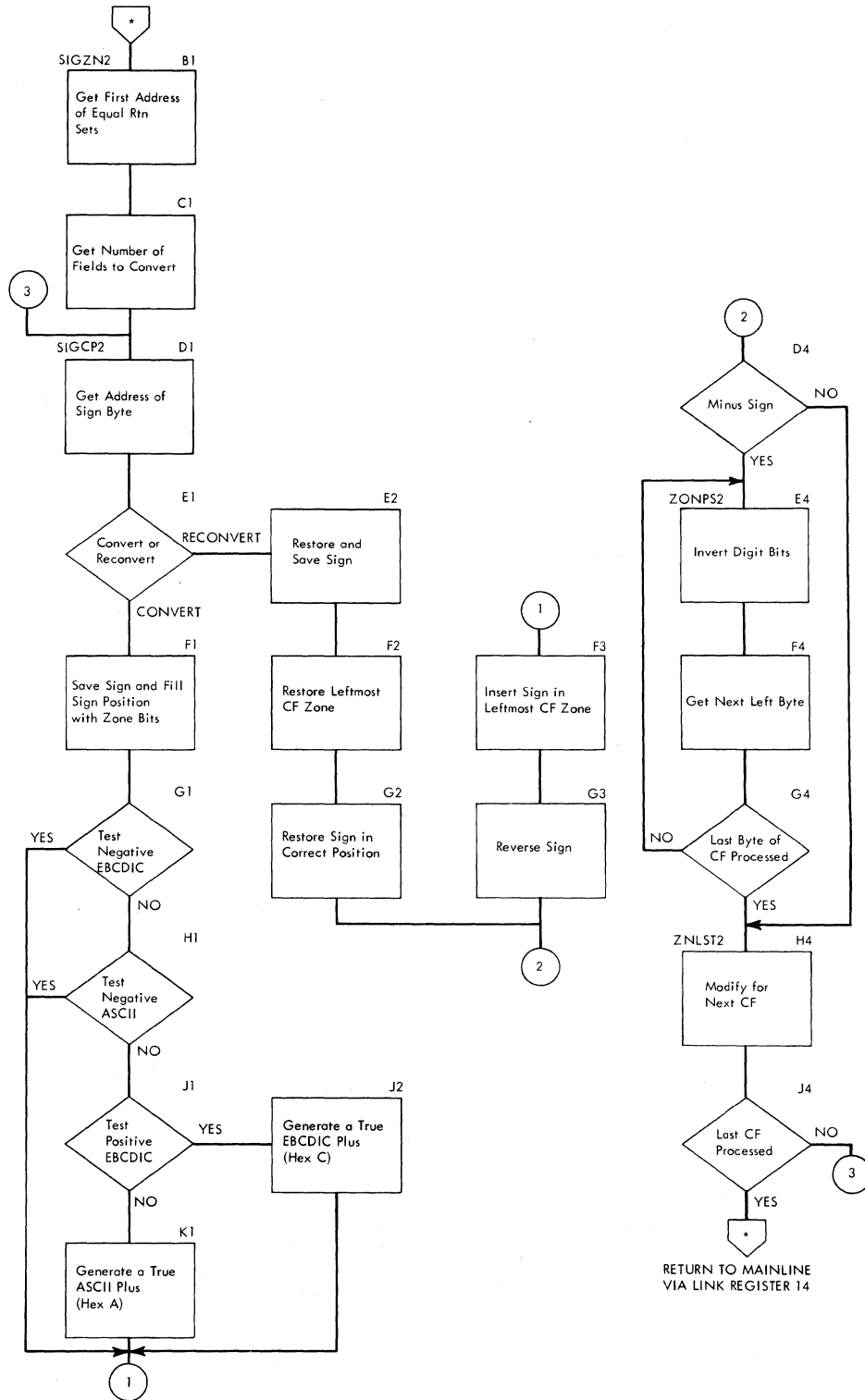


Chart FE. Zoned-Decimal (SIGZON) Convert/Reconvert

BAL FROM MAINLINE
 (PHASES 1 AND 3 FOR SORT,
 PHASE 4 FOR MERGE-ONLY)

NOTE: Convert in
 phases 1 and 4.
 Reconvert in
 phases 3 and 4.

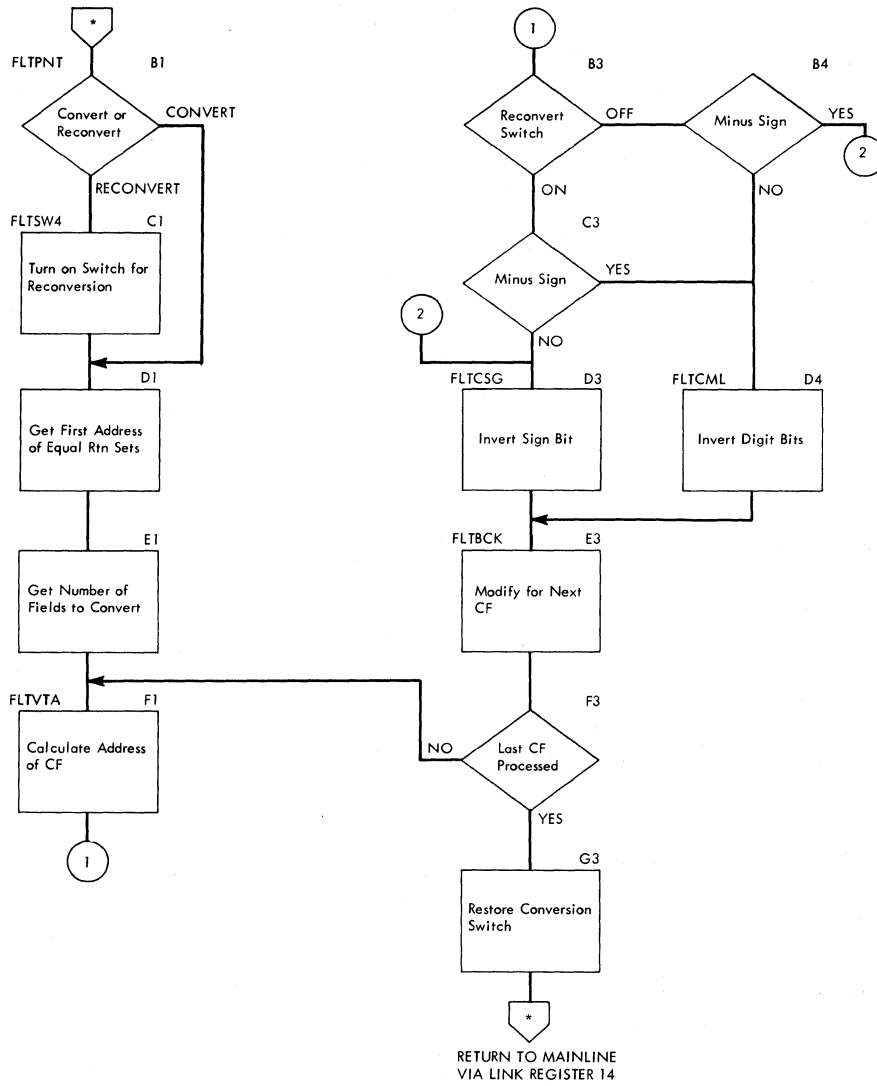


Chart FF. Equal Routine, Phases 2, 3, and 4

CONSTANTS

ASSIGNMENT PHASE CONSTANTS

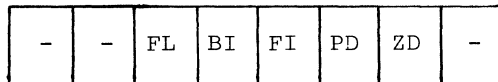
<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
ADD1	Starting address of user routines in phase 1	4
ADD34	Starting address of user routines in phase 3 or 4	4
ADDLUB	Address of logical unit block	2
ADDPUB	Address of physical unit block	2
ADDR0T	Message: "ADDR0UT="	8
ADDR0UT	Field definer: "ADDR0UT"	7
ADDWD	Location of control card in TBLADD	4
ADDWD1	Length of logical control card in TBLADD	2
ADDWD2	Code of control card in TBLADD	2
AL3	Decimal zero	1
ANS1 } ANS2 } ANS3 }	Calculated results from TABLEF, using TABLE, TABLE2, and TABLE3 respectively	12
ASAVE	Address constant used to locate storage area, SAVE	4
AVAILC	Available storage for input, output, and work areas	4
B	Computed internal sort blocking factor	2
B3MAX	Calculated maximum phase 3 (output) block length	4
BI	1. Format code designating use of unsigned binary 2. Calculated number of records per input block from overlay 9 (DSORT009)	2 4
BLANKS8	Two full words of blanks (Hex 40)	8
BLK123	Three 4-byte temporary storage areas for calculated value of BMAX for phase 1, 2, and 3 respectively	12
BLKSIZ	Input blocksize given by user	2
BLKSOZ	Output blocksize (maximum blocksize for variable-length records). This area is also used to store L5 during scanning.	2
BLKSIZE	Field definer: "BLKSIZE"	7
BMAX	Computed maximum block length used by all program phases	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
BO	Output blocking factor calculated by overlay 9 (DSORT009)	4
BPT	Calculated number of sort blocks per 2311 track	4
BPT1IP	Calculated phase 1 input blocks per track for fixed-length records	2
BPT3OP	Phase 3 output blocks per track for fixed-length records	2
BYBKLS	Computed number of bytes per last block	2
BYPAS	Field definer: "BYPASS"	6
BYPSSA	Message: "BYPASS"	6
BYTBLK	Calculated maximum number of bytes per block, including disk gaps.	2
BYTECT	Byte count of save area	4
C10	Character constant 10	2
C11	Character constant 11	2
C20	Character constant 20	2
C31	Character constant 31	2
C41	Character constant 41	2
C101	Character constant 101	3
C401	Character constant 401	3
CALCAR	Message: "CALCAREA"	8
CALCAREA	Field definer: "CALCAREA"	8
CANCEL	Lower case compare constant for "cancel" reply	6
CANCELUP CANCPCAP }	Upper case compare constants for "CANCEL" reply	6
CCAREA	Main storage area containing compressed unsorted control cards	736
CCB1	CCB for printing messages on SYSLOG or SYSLST	16
CCBRCP	CCB for reading checkpoint for RESTART	16
CCBWCP	CCB for writing checkpoint on disk	16
CCDS	Message: "*** CONTROL CARDS ***"	24
CCEND	Address of end of message area	4
CCSADD	Address of start of current message	4
CCSAVE	Save area for sorted compressed control cards and unprinted messages	736
CCW1	CCW for printing messages on SYSLOG or SYSLST	8

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
CCW1+8	CCW for reading operator reply if SYSLOG is a 1052	8
CDCOUNT	Control card count	1
CF1LCT through CFCLNG }	Location of control fields 1 through 12	24
CF1SEQ through CFCSEQ }	Collating sequence of control fields 1 through 12	24
CFA	Save area for control field A computation used to check for control field overlap	2
CFB	Save area for control field B computation used to check for control field overlap	2
CFPL10	Total length of control fields plus 10 (TLACFD +10)	2
CIMAGSV	Address of next control card save area (when all control cards have been read, contains the address of the first byte past the save area used)	4
CKPT	Checkpoint disk address (DDCHHR)	6
CLOSE	Field definer: "CLOSE="	6
CONBKT	Indicator for heading printed	1
CONHDG	Message: "*** COMPUTED CONSTANTS ***"	24
CONSTANT	Heading to be printed for printout of calculated constants - see program listing	83
CORESZ	User given or actual machine size	4
CORHLD	Main storage size from system generation time	4
CP2LP8	Denominator [2(L1)+8] used in calculation of PH1B1	2
CPST	Number of cylinders required to accommodate a phase 1 sequence	4
CTR	Count of the number of pointers to work area initialized	1
DATE	Current data given by user	8
DATECONV	Date convention used at creation time: 00000000 - MMDDYY 00000001 - DDMMYY	1
DEVTAB	A maximum of seven 4-byte entries. The first 2 bytes of each entry contain a device number obtained from TABLEB. The second 2 bytes contain the total number of available tracks on the device.	28 max
DOUBLE	Phase work area for conversion routines	32

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>								
DUPBK	Eight-byte save area for processed control card codes: 1 - SORT 5 - OUTFIL 2 - MERGE 6 - MODS 3 - RECORD 7 - OPTION 4 - INPFIL 8 - END	8								
E03--E06 E16--E23 E25--E27I E28--E49 E51--E54 E57--E61 E64--E85 E90--E92 E3050	Error messages - see program listing and Appendix D.									
E06BKT			Error "7D06I" indicator - see Appendix D.	1						
E68WRK	Work area used to reinitialize error message E68 "7D68I" (see Appendix D).	59								
END	Statement definer: "END"	3								
ERRBK	Error indicator: OFF - hex 0 ON - hex D2	1								
EX1	Byte 0 - Phase 1 exit indicators	2								
	<table border="1" style="margin-left: 40px;"> <tr> <td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>EXIT 13</td><td>EXIT 12</td><td>EXIT 11</td> </tr> </table>	-	-	-	-	-	EXIT 13	EXIT 12	EXIT 11	
-	-	-	-	-	EXIT 13	EXIT 12	EXIT 11			
	Byte 1 - Block type and tape options									
	<table border="1" style="margin-left: 40px;"> <tr> <td>VAR BLK</td><td>FIXED BLK</td><td>OPEN RWD</td><td>OPEN NORWD</td><td>-</td><td>CLOSE UNLD</td><td>CLOSE RWD</td><td>CLOSE NORWD</td> </tr> </table>	VAR BLK	FIXED BLK	OPEN RWD	OPEN NORWD	-	CLOSE UNLD	CLOSE RWD	CLOSE NORWD	
VAR BLK	FIXED BLK	OPEN RWD	OPEN NORWD	-	CLOSE UNLD	CLOSE RWD	CLOSE NORWD			
EX34	Phase 3 or 4 exit indicators	1								
	<table border="1" style="margin-left: 40px;"> <tr> <td>-</td><td>-</td><td>-</td><td>EXIT 45</td><td>EXIT 44</td><td>EXIT 43</td><td>EXIT 42 or 32</td><td>EXIT 41 or 31</td> </tr> </table>	-	-	-	EXIT 45	EXIT 44	EXIT 43	EXIT 42 or 32	EXIT 41 or 31	
-	-	-	EXIT 45	EXIT 44	EXIT 43	EXIT 42 or 32	EXIT 41 or 31			
EXIT	Message: "EXIT"	4								
F12	Constant 12	4								
F512K	Constant 524288 used to determine limit of main storage size	4								
F4096	Constant 4096 used as relocation factor for overlay 1 (DSORT001)	4								
F16384	Constant 16384 used to test for storage size of 16K	4								
FI	Format code: FI - fixed point	2								

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
FIELDS	Field definer: "FIELDS"	6
FILEDC	Field definer: "FILES"	5
FILEMS	Constant allows all disk writes	1
FILES	Number of input files given by user (see MRGVOL)	1
FILESI	Field definer: "SIZE"	4
FILESZ	Number of records to be sorted; given by user	4
FIOCS	Time required to execute IOCS; determined by machine model	4
FIXED	Field definer: "FIXED"	5
FL	Format code: "FL" - floating point	2
FMATEK	User given format code for PRINT option	2
FNP	Number of merge passes calculated for run time in overlay 9 (DSORT009)	4
FORM	Phase 1 format time calculated in overlay 9 (DSORT009)	4
FORMAT	1. Byte 0 - 8 bits for format indicators	2



FL - floating point
 BI - unsigned binary
 FI - fixed point
 PD - packed decimal
 ZD - zoned decimal

Byte 1 - NOCFLD, number of control fields in binary

	2. Message: "FORMAT" used in overlay 9 (DSORT009)	6		
FRACT	Fraction (or portion) of sort to be executed on one 2311 disk drive used in timing calculations in overlay 9 (DSORT009)	4		
G	Computed number of records in a phase 1 sequence	4		
GA1	Available core for sort	4		
GA2	Computed number of sort blocks, plus an 8-byte tag for each record in the sort block, in GA1	2		
H0 through H12	Hex constants (2 bytes each). 0 through 10 used for overlay numbers.	26		
H25			Constant 25	2
H80			Constant 80	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
H255	Constant 255	2
H256	Constant 256	2
H300	Constant 300	2
HALFTK	Half of the user-given work area tracks minus checkpoint tracks.	2
HEAD	Heading used for print out of modifications, name, address, and exit number.	50
HL8 through HL25X }	Nine half-word literal constants--see program listing of assignment phase overlay 9 (DSORT009) for specific values.	18
HOLDC HOLDCY }	Total number of cylinders in TABLEF.	4 4
IBC	Sort block size in number of characters (bytes) used in run time calculations.	4
IBIC	Input block size in number of characters (bytes) used in run time calculations.	4
IBOC	Output block size in number of characters (bytes) used in run time calculations.	4
IBR	Number of records in a sort block, used in run time calculations.	4
IGNORE	Lower case character constant used to check operator reply "ignore".	6
IGNOREUP	Upper case character constant used to check operator reply "IGNORE".	6
INBSV	Input print heading for: INPUT, BLOCKSIZE, TYPE, TAPE OPEN, and CLOSE.	67
IND85	Error "7D85I" indicator (see Appendix D).	1
INP	Indicator used in run time calculations to determine odd or even number of passes.	4
INPFIL	Statement definer: "INPFIL"	5
INPUT	Field definer: "INPUT="	6
INPUTMRG	Byte 0 - Output file Byte 1 - Input FILEA Byte 2 - Input FILEB Byte 3 - Input FILEC Byte 4 - Input FILED Byte 5 - INPBK Byte 6 - SYSLGBIT Byte 7 - SKIPCODE	8

Label

Contents

Bytes

Byte 0

—	UNLAB	NON STD	STD	—	DISK OUTPUT OLVP	DISK OUTPUT	TAPE OUTPUT
---	-------	------------	-----	---	------------------------	----------------	----------------

Byte 1 through 4 (FILEA through FILED)

—	UNLAB	NON STD	STD	—	—	DISK INPUT	TAPE INPUT
---	-------	------------	-----	---	---	---------------	---------------

Byte 5 (INPBK)

—	—	—	—				
---	---	---	---	--	--	--	--

TYPE OF INPUT

- 0001 - Tape only
- 0110 - Disk only
- 0111 - Mixed tape
and disk

Byte 6 (SYSLGBIT) -
 Bits 0 through 5: unused
 Bit 6: SYSLOG = SYSLST
 Bit 7: SYSLOG = 1052

Byte 7 (SKIPCODE) - DTFCP carriage control byte

INQCCB	CCB for operator communication write	16
INQCCW	CCW for operator communication write	8
INQCCW+8	CCW for reading operator communication reply if SYSLOG is a 1052	8
INTPRL	Four 2-byte entries for L1, L2, L3, and L4.	8
	L1 (INTPRL) - input record length (fixed-length records) or maximum input record length (variable-length records).	
	L2 (INTPRL+2=NOTUSE) - not used by sort program.	
	L3 (INTPRL+4=OTPTRL) - output record length (fixed-length records) or maximum output record length (variable-length records).	
	L4 (INTPRL+6=MINRCL) - minimum record length (variable-length records only).	
IRL	Calculated sort record length used in run time calculations.	4
IRPT	Calculated number of records per 2311 track used in run time calculations.	4
JOBNAME	Job name as it appears in job control statement.	8


<u>Label</u>	<u>Contents</u>	<u>Bytes</u>																
KEY	Byte 0 - keylength, Byte 1 - other options. Byte 1	2																
	<table border="1"> <thead> <tr> <th>TAG TYPE</th> <th>CALC AREA</th> <th>N</th> <th>V</th> <th>B</th> <th>P</th> <th>R</th> <th>C</th> </tr> </thead> <tbody> <tr> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </tbody> </table>	TAG TYPE	CALC AREA	N	V	B	P	R	C									
TAG TYPE	CALC AREA	N	V	B	P	R	C											
	TAGTYPE: 0-tag only, 1-tag plus control data.																	
	CALCAREA: 0-option not specified, 1-option specified.																	
	N - NOTPMK option																	
	V - VERIFY option (disk only)																	
	B - BYPASS (tape input for sort, tape input or output for merge).																	
	P - PRINT option.																	
	R - ADDROUT option (disk input only).																	
	C - RESTART option (sort only).																	
KELEN	Field definer: "KEYLEN"	6																
KEYRTN	Message: "KEYLEN"	6																
L2 through L3624X }	37 full word literal constants used in run time calculations--see program listing of overlay 9 (DSORT009) for specific constant and value.	148																
L3MAX	Computed phase 3 (output) maximum record length.	4																
L5	Average length of a variable-length record without tag. (Note: L5 is used only for a sort run when variable-length records with no ADDROUT has been specified.)	2																
LABBKT	Save area for label information.	5																
LABEL	Field definer: "LABELS"	6																
LBND	Lower pack boundary used in run time calculations.	4																
LBSW	Retains label information in one byte. Bits 0 through 3 - Output labels Bits 4 through 7 - Input labels 0001 -- standard labels 0100 -- unlabeled 0010 -- non-standard labels	1																
LEN1	Length of phase 1 exit routines.	4																
LEN34	Length of phase 3 or 4 exit routines.	4																
LENGTH	Field definer: "LENGTH"	6																
LF2	Hex constant 000000F2	1																

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>		
LGTH	Message constant for printing values of L1 through L5.	50		
LL1 through LL4	Assumed disk work area limits for calculating run time.	16		
LLS			Line heading for printout of location, length, sequence, and format of control fields.	47
LLSITK			Save area for computed total number of lower limit tracks used in run time calculations.	4
LMAX			Calculated maximum record length used by phases 1, 2, and 3.	4
LMAXNO	Constant "7FFFFFFF" used in run time calculations.	4		
LOC	Location of control fields for print messages.	16		
LOGWKA	Two 2-byte entries containing location of pointers indicating start of each logical area in TABLEB.	4		
LOWTM	Save area for lowest calculated sort time for trial order of merge in run time calculations.	4		
M	Value of denominator $[6(L1/256)]$ used in phase size calculations.	2		
MANDBK	Eight 1-byte entries for retention of code for each logical control card processed.	8		

S	M	R	I	O	E	O	D
---	---	---	---	---	---	---	---

S=SORT O=OUTFIL
M=MERGE E=MODS
R=RECORD O=OPTION
I=INPFIL D=END

MAXBL	Calculated maximum block length that can be processed by phases 1, 2, and 3.	4
MAXBL1 MAXBL2 MAXBL3	Calculated maximum block length for phases 1, 2, and 3 respectively.	12
MERGE	Statement definer: "MERGE"	5
MF	Message: "MERGE FIELD"	11
MFS	Computed maximum file size (number of records to be sorted) for a sort run only and fixed-length or ADDROUT records.	2
MID	Midpoint of disk work area used in run time calculations.	4
MODS	Statement definer: "MODS"	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>										
MRGVOL	Number of volumes and files for sort.	10										
	FILE A B C D E F G H J Number of volumes per file <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> </tr> </table> <div style="text-align: center;">  <p>Total number of files (MRGVOL+9)</p> </div>											
MS	Machine size either given by the user or extracted from the communications region.	4										
MSG01 through MSG78	Print messages; see program listing for specific message.											
MSGADD	Main storage address of control card being processed.	4										
MSGI1	Message: "INFIL"	6										
MSGR2	Message: "RECORD"	6										
MSGSM1	Message: "7D08I" (see Appendix D).	54										
MXBYPT	Calculated maximum number of bytes per 2311 track.	2										
N	Number of control fields (NOCFLD) plus 10 used in phase size calculations.	2										
NOCFLD	Number of control fields--in binary (equates to FORMAT+1).	1										
NONE	Message: "NONE"	4										
NONST	Message: "NONST" (non-standard labels)	5										
NOP	Hexadecimal 4700 used to prevent execution of a branch instruction.	2										
NOPASS	Computed number of merge passes in phases 2 and 3	2										
NOPMSG	Message: "OPTIONS SPECIFIED"	17										
NORWD	Message: "NORWD" (no rewind)	5										
NOSEQ	Computed number of output sequences from phase 1.	4										
NOTPMKL	Field definer: "NOTPMK" (tapemark option)	6										
NP	Computed number of merge (phase 3) passes used in run time calculations.	4										
NRTHWK	Number of tracks in half work area computed by run time calculations (DSORT009).	4										
NRTOUT	Number of tracks required for output computed by run time calculations (DSORT009)	4										
NRTRKS	Total number of tracks required (NRTHWK+NRTIN+NRTOUT).	4										
OM	1. Computed order of merge for a sort run or given order of merge for a merge run--see FILES.	2										

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
	2. Trial order of merge used in run time calculations in overlay 9 (DSORT009).	4
OMMSG	Message: "FILES"	5
OMPL1	Order of merge plus 1.	2
ONE	Full word constant 1 used in run time calculation (DSORT009).	4
OPEN	Field definer: "OPEN"	5
OPERRBK	Indicator for operator intervention error (same conditions as ERRBK).	1
OPMSG	Message: "OPTION"	30
OPTION	Statement definer: "OPTION"	6
OTWKAR	Message: "OUTPUT IN WORKAREA"	18
OUTFIL	Statement definer: "OUTFIL"	6
OUTMS	Message: "OUTPUT BLOCKSIZE XXXXXXXX LABEL XXXXXX"	30
OUTPUT	Field definer: "OUTPUT="	7
PATCH	1. 44-word patch area in overlay 2	176
	2. 44-word patch area in overlay 6	176
PD	Format code "PD"-packed decimal.	2
PH1B1	Calculated maximum number of records in a phase 1 sort block.	2
PH1B2	Result of alternate calculation of the maximum number of records in a phase 1 sort block.	2
PHAS1	Field definer: "PH1="	4
PHAS3	Field definer: "PH3="	4
PHAS4	Field definer: "PH4="	4
PHAS34	Symbolic name for phase 3 or 4 exit routines.	8
PHASE1	Symbolic name for phase 1 exit routines.	8
PHS1 } PHS2 } PHS34 } PHS4 }	Minimum sizes for indicated phases of the program.	8
PHS2MIN } PHS3MIN }	Calculated minimum size for phases 2 and 3 respectively.	4
PHS2MX } PHS34MX }	Computed maximum size for phase 2 and 3, respectively, for OM 5 to 7 (fixed-length records) or 4 to 6 (variable-length records).	4
PIOCS	XTENT retrieval area at FILEW open time	72
PRINT2	Message: "PRINT"	5
PRTMSG		5

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
PRNT	Field definer: "PRINT"	5
PROC1 } PROC2 }	Process time for phases 1 and 2, respectively, computed by run time calculations (DSORT001).	8
R3SAVE } R4SAVE } R5SAVE } R8SAVE } R9SAVE } RASAVE }	Storage area for general registers 3,4,5,8,9, and 10 respectively.	24
R7SAVE	Double word storage area for two counter registers used in check of control field overlap.	8
RAFRTN	Message: "ADDROUT"	7
RBBKT	Save area for control card codes.	2
RDT1	Computed phase 1 read time.	4
RDWR23	Read and write time calculated by DSORT009 for phases 2 and 3.	4
READCK	CCW for reading checkpoint count, key, and data	8
READIN	Read-in and write-out area for print CCW.	100
RECORD	Statement definer: "RECORD"	6
RECTYP		1

V	F	OPEN RWD	OPEN NORWD	—	CLOSE UNLD	CLOSE RWD	CLOSE NORWD
---	---	-------------	---------------	---	---------------	--------------	----------------

V - variable-length records
F - fixed-length records
RWD - rewind
NORWD - no rewind
UNLD - unload

RESTAT	Field definer: "RESTART"	7
RESTRT	Message: "RESTART"	7
RETRY	Lower case constant used to check operator reply "retry."	5
RETRYUP	Upper case constant used to check operator reply "RETRY".	5
RPTBKT	Save area for calculated number of records per 2311 track.	2
RT	Message: "RECORD TYPE"	11

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
RUNCOD	User run code from columns 73-80 of first control card (for assignment phase only)	8
RWIND	Message: "RWD" - rewind.	3
S	Size, length and collating sequence of control cards for printout.	21
SAVE	Save area for calculated constants computed for each internal order of merge of a sort run. Used for run time analyses, SAVE contains 6 sets of: OM (internal order of merge) - 2 bytes B (number of records per internal sort block) - 2 bytes SBSIZE (internal sort block size in bytes) - 2 bytes BPT (maximum number of internal sort blocks per 2311 track) - 2 bytes G (number of records per phase 1 sequence, or maximum number of records for variable-length with no ADDR0UT) - 4 bytes GA1 (available storage for sort) - 4 bytes GAREA (number of bytes in a phase 1 sequence, excluding tag area, in GA1) - 4 bytes GAREA1 (number of bytes in a phase 1 sequence, including tag area, in GA1 -- or, average number of records in a sequence for variable-length with no ADDR0UT) - 4 bytes NOBLKG (number of sort blocks in GAREA) - 2 bytes MXBYPT (maximum number of bytes per 2311 track -- becomes relative time after DSORT009 has been processed) - 2 bytes AMAX (Maximum work area size) - 2 bytes NOPASS (number of merge passes in phases 2 and 3) - 2 bytes <u>Note:</u> The routine beginning at the label PRTEXIT in overlay 10 (DSORT010) will print the contents of SAVE for diagnostic purposes. Refer to the listing comments for an explanation of the use of the PRTEXIT routine.	192
SAVERA	Save area for link register (general register 10).	4
SAVERERF	Save area for general registers 14 and 15.	8

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
SAVIMAG	Storage area used to retain control card image (up to a maximum of 25 cards) for print.	2004
SAVREGS	Temporary storage used to retain contents of miscellaneous registers.	20
SAVREM	Save area for the remainder, during a divide instruction, resulting from calculations of LMAX and BMAX.	2
SAVROM	Pointer to trial order of merge used in run time calculations.	4
SAVRPT	Number of records per 2311 track for trial order of merge.	4
SBSZPTAG	Denominator: $SBSIZE + [8(SBSIZE/L1)]$, used in the calculation of GA2 (number of sort blocks, plus an 8-byte tag for each record in the sort block, in GA1).	2
SECCTR	Number of sections into which the work area is divided.	1
SEQ	Collating sequence of control fields 1 through 12.	12
SF	Message: "SORT FIELD"	10
SHTSK2	Shortest phase 2 seek time computed by run time calculations in DSORT009.	4
SIZE	Field definer: "SIZE"	4
SKLEN	Calculated length of seek in number of cylinders.	4
SKT	Calculated time for one seek.	4
SKT2	Total phase 2 seek time computed by run time calculations in DSORT009.	4
SKT3	Total phase 3 seek time computed by run time calculations in DSORT009.	4
SORMBK	S-character (sort) or M-character (merge) used to indicate type of run.	1
SORT	Statement definer: "SORT"	4
SPDFAC	Speed factor determined by machine model.	4
STD	Message: "STD"	3
STORAG	Message: "STORAGE="	8
STORED	1. Save area for general registers 5 through 10 in CCERR routine.	24
	2. Save area for general registers 2 and 3 in overlay 7 (DSORT007).	8

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
STP123	Calculated values of LMAX for phases 1, 2, and 3 respectively.	12
STRING	Computed number of output sequence from phase 1.	4
SUBTRC	Number of bytes, plus 1, in current statement definer.	4
SUPER	Actual supervisor program size.	4
SVBK	Current message length.	4
SYSDATE	System creation date.	6
SYSTBLE	Ten 4-byte entries for SYS001 through SYS010 plus a 4-byte end-of-table indicator. Byte 0 - Channel number Byte 1 - Unit number Byte 2 - Device type Byte 3 - Logical unit (SYS0nn)	44
TA	Number of tracks available as given by XTENT cards.	2
TABLE TABLE 2 } TABLE 3 }	Three 72-byte tables for the determination of ANS1, ANS2, and ANS3, respectively, for orders of merge 2 through 7.	226
TABLEB	Disk work area limits table.	88

SECT	UN#	LOWER LIMIT				UPPER LIMIT			
		C	H	H	R	C	H	H	R
0000									
000C									
0078									
0024									
0030									
0030									
0048									
0000	0000								

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
TBLADD		64
	<p>Eight 4-byte entries:</p> <p>A=2-byte address of first byte of compressed control card in CCSAVE.</p> <p>B=total number of bytes in compressed card.</p> <p>C=code indicating type of card</p> <p>1=SORT, 2=MERGE, 3=RECORD, 4=INPFIL, 5=OUTFIL, 6=MODS, 7=OPTION, 8=END</p> <p>This is an internal record of cards processed. It is used to get the control card, if necessary, for a message printout and for scanning control cards.</p>	
TIME	Total run time computed by trial order of merge.	4
TLACFD	Total length of all control fields.	2
TR	Number of tracks required in logical work area.	2
TRACK	Total number of tracks in work area, given by user extent cards.	2
TROVER	Number of tracks not yet used in building TABLEB	2
TRUNDR	Total number of tracks still required in building TABLEB	2
TYPE	Field definer: "TYPE"	4
UCYHWK	Number of cylinders used in half work area computed by run time calculations DSORT009.	4
ULLBK	XTENT lower limit, in tracks.	2
UNLAB	Message: "UNLAB" (unlabeled)	5
UNLD	Message: "UNLD" (unload)	4
UULBK	XTENT upper limit, in tracks.	2
VAL1	Save area for first value calculated in TABLEF.	4
VARBLE	Message: "VARIABLE"	8
VERFY	Field definer: "VERIFY"	6
VOLMG	Message: "INPUT FILEA FILEB FILEC FILED FILEE FILEF FILEG FILEH FILEI"	68
VOLUM	Field definer: "VOLUME"	6
VOLUME	Message: "VOLUME="	7
WORD	Total number of work-area tracks computed by run time calculations in DSORT009.	4
WRITE	CCW for writing checkpoint record	8

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
WRT	Phase 1 read and write time in excess of phase 1 read time (RDT1) computed by run time calculations.	4
XTADDR	Pointer to current XTENT card location.	4
XTAREA	Storage area for up to a maximum of 6 converted XTENT cards:	84



14-Byte Entries for each XTENT:

- A = type - 1 byte
- B = sequence - 1 byte
- C = lower limit (CCHH) - 4 bytes
- D = upper limit (CCHH) - 4 bytes
- E = class and unit - 2 bytes
- F = old bin number - 2 bytes

XTENT	XTENT card read-in area.	12
ZEROS	Six full words of zeros.	24
ZD	Format code "ZD" - zoned decimal	2


PHASE 1 CONSTANTS

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
ADR2	Location of disk address (DSKADR2).	4
ADRBKT	Disk address in CCHH format.	4
ADSAVE	Starting address of current string.	4
ADTAB1	Location of address table 1.	4
ALPHA	Constant 8.	2
ARNYAD	Starting address of conversion routine.	4
BLKSZ	Input block size from assignment phase; given by user.	2
BLKSZ1	Maximum input block size for variable-length records.	2
BR4SAV	Save area for RR format branch.	4
CBYTE32	DTF byte 32 for CLOSE rewind/no rewind option.	1
CCB2	Output CCB.	16
CCBPR	System log CCB.	16
CCW1	System log CCW.	24
CMPL10	Complement of 10 (0000FFF6).	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
CMPADR	Starting address of compare string in chain.	4
CMPEND	End address of compare chain.	4
CMPLT	Complement of track number 10 (00FFF600-CHHR); used to calculate track number of next cylinder.	4
CMPOF8	Complement of track number (00FFF800-CHHR).	4
CMPOF9	Complement of track number 9.	4
DALIM	Disk input area upper limit (CCHH format).	4
DCCWSI	Loop CCW for read input.	16
DCCWSK	Seek/search CCW for read input.	16
DEVICE	Disk drive address.	2
DIKEYL	Key length (disk).	4
DINABK	Disk address of search argument of read.	5
DINADR	Address of current input block.	8
DINCCX	Read count CCB.	16
DINCCY	Read input CCB.	16
DINSEK	Disk address in BB CC HH R format. BB = bin number (always 00) CC = cylinder number HH = read/write head R = number of the data record on the track	8
DINXAD	Address of next input block.	8
DNBTK	Seek/search address for 1 block per track.	8
DRCNT	Work area for pack/unpack routines.	8
DSAVE1	Save area for level 2 register	8
DSAVE2	Save area for move registers.	12
DSAVE3	Save area for move constants (base registers G and B).	12
DSBKT	Starting address of DSHIFT table.	4
DSEEK	Output seek CCW.	8
DSHIFT	Output control table used for storing starting disk addresses of each sequence in a set. Consists of up to 8 sections of 12 bytes each. See output routine narrative for detail description (Chart BG).	96 max.
DSKADR1	Disk address in BB CC HH R D format: BB - bin number (always 00) CC - cylinder number (0-199) HH - head number (0-9) R - record number D - unused	8

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
DSKADR2	Disk address in BB CC HH R K DL format: BB - bin number (always 00) CC - cylinder number HH - head number R - record number (displacement) with track K - keylength DL - data length	10
DSKLIM	First 4 bytes - address of input area upper limit. Second 4 bytes - address of input area lower limit.	8
DSKTBL	Disk number table (0 through 7, two bytes each).	16
DSPLMT	Displacement, in bytes, of current record in input block.	2
DSRCH	Output search CCW.	16
DWRITE	Write output block CCW.	8
EIGHT	Constant 8.	2
ENDADDR	Starting address of EOFADR (end-of-file routine).	4
ENDPRG	Indicator for end of program.	1
EOFADR	Location of end-of-file routine.	8
EOFIND	End-of-file indicator.	1
EXITCW	Seek/search checkpoint address CCW.	16
FIBKT	Upper main storage address of last record in a file.	4
FILENAME	File identification (FILEA).	8
FILES	Constant ABCDEFGHI used in FILENAME.	9
FIX2A5	Save area for general registers 2,3,4 and 5 in fixed-point conversion routine.	16
FLT2A6	Save area for general registers 2,3,4,5 and 6 in floating-point conversion routine.	20
FLTCP2	Hexadecimal F's used in floating-point conversion routine.	8
FSEXC	Message: "7DA3I" (see Appendix D).	40
HICKY	Starting address of input area in main storage.	4
IALIM	End address of input area in main storage.	4
IBVA04	Location of output area in main storage.	4
IBVA05	Location of address table 2.	4
IBVCOMP	BPT (blocks per track) complement.	2
IBVLV1	Starting address of input area.	4
IBVNR4	Constant 4.	2
IBVNRB	Number of sort blocks in "G".	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
IBVNRV	Number of input volumes.	2
IBVODM	Order of merge.	2
IBVSET	Number of blocks in a set ("set" is defined as the number of strings or sequences equal to the order of merge).	2
IBVTAB	Table for storing number of blocks per string in last pass. See text for chart BH, end-of-phase routine, for detailed explanation of use.	16
IBVTOT	Storage area for total block count.	4
IFOR1 through IFOR7 }	Used in checkpoint address for conversion routines.	48
INAREA	Starting address of input area.	4
INLIM	End address of input area.	4
INMAX	End of current input area.	4
INTBYP	Storage area for number of records bypassed.	4
INTMES	Message: "7DA2I" (see Appendix D).	42
KEYLNG	Save area for key length.	2
L1CXA3	Current record address (variable-length records).	4
L1CXA4	Location of address table 1 (variable-length records).	4
L1CXA5	End address of input area,	4
L1CXA6	Address of RR format branch.	4
L1CXA7	Upper limit of current block.	4
L1CXA8	Location of current record count.	4
LALADR	Checkpoint seek/search disk address.	8
LBLBASE	Starting address of phase 1, used to initialize base registers.	4
LBLCCB	Checkpoint seek CCB.	16
LBLCK1	Internal sort CCW.	8
LBLSEK	Seek/search CCW.	16
LSTH	Read/write checkpoint CCW.	8
MASK10	Linkage from mainline to move routine.	4
MAXFSZ	Maximum work area size.	1
MAYDAY	Upper limit of current variable-length block.	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
MPASS	Message: "7DA5I MERGE PASSES ".	19
NBRVOLS	Number of input volumes and files (equivalent to assignment phase MRGVOL): Volumes/file <div style="text-align: center;">  <p>File A B C D E F G H I ↑</p> </div> <p style="text-align: center;">Number of files (NBRVOLS + 9)</p>	10
NC3BKT	Save area for read or write address in output routine.	4
NCFBKT	Number of input control fields.	2
NOBG	Number of blocks in "G".	4
NOUTM	Constant (00FFFFFF).	4
NR003	Constant 3.	2
NR02	Constant 2.	2
NR10	Constant 10.	2
NRBLM	Number of blocks in last order of merge.	4
NRBLS	Number of blocks in last sequence.	4
NRDUB	Location of number of doublets to be processed for level 1 sorting.	4
NRPAS	Number of passes for merge phases.	2
NRSEQ	Number of sequences.	4
OBYTE32	DTF byte #32 for various OPEN options.	1
OM2	Alternate OM bucket used in exit routine.	2
ONEXID	Relocation factor for variable-address routine.	2
OUTRD	Save area for compression routine read.	8
OUTWT	Save area for compression routine write.	12
OV2BKT	Save area for general register 5 in end-of-phase routine.	4
OV2END	Save area for general register 4 in end-of-phase routine.	4
P1BASE	Address of start of subroutine for building doublets for level 2.	4
P1END	Start of checkpoint record.	8
P1G	"G" count (number of records in input area).	4
P2CCB	Read/write checkpoint CCB.	16
P2TBL	Checkpoint disk address (BB CC HH R).	8

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
P3SVRG	Save area for general registers 2 and 3 in read checkpoint routine.	8
PAKRG2	Save area for general registers 4 through 9 in SIGPAK conversion routine.	24
PASNR	Pass number for checkpoint routine.	2
PATCH 0 through PATCH 7 }	Areas reserved for possible patch routine.	112
PB2BKT	Number of blocks in last set.	2
PHEND	Message: "7DA6I END PHASE 1".	17
PKSGN2	Save area for sign in SIGPAK conversion routine.	2
RAFEND	Address of current RAF end.	4
RAFILE	RAF start address.	4
RCDBKT	Number of records processed; used in end-of-phase routine.	4
RDCNT	Number of records processed.	4
RCDLEN	Record length.	4
RCDPR	Message: "7DA4I RECORDS PROCESSED".	24
RECLNG	Original record length.	2
SAVREG	First 4 bytes - record address. Second 4 bytes - location of block size count. Last 4 bytes - location of record displacement.	12
SAVRG14	Save area for general register 14 in initialization of multi-volume routine.	8
SORTRL	Save area for sort record length.	2
TAPLUA	Log unit address.	2
TCCWSK	Work area used in read and convert routines.	8
TINADR	Disk address of current input block.	6
TINLST	Relocation factor of disk-input routine.	3
TINMUA	Relocated address of call for additional-volume routine.	4
TOTCFI	Total length of input control fields.	2
UPADDR	Location of address table 1.	4
UPLIM	Upper limit of file on disk.	4
USEAD2	User routine starting address for tape input.	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
USERAD	User routine starting address for disk input.	4
USADR	Location of user routine.	4
VBART	Work area used in input-finish, build-address, and compute-areas subroutines.	4
VLORST	Save area for starting address of output area in variable-length output routine.	4
VLOSCN	First 4 bytes - current address in output area. Second 4 bytes - output area upper limit.	8
WKBKT	Work area used in exit routine	8
WKTAB	Work area limit table obtained from assignment phase TABLEB through the checkpoint record.	88
WLRBKT	Record length used for WLR check.	2
WLRCCW	CCW for Print WLR message.	8
WLRMES	Message: "7DA1I" (see Appendix D).	15
WRKBKT	Work area.	4
WRKCCB	Work area error print CCB.	16
WRKCCW	Work area error print CCW.	8
ZERO	Constant 0.	2
ZNPAS	Number of merge passes.	4
ZONRG2	Save area for general registers 5 through 8 in SIGZON routine.	16
ZONZN2	Storage area for signs in SIGZON routine.	2
ZRCNT	Number of record for print routine.	8
ZROMSK	Zero mask.	14

PHASE 2 CONSTANTS

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
ABEGIN } through GBEGIN }	Starting addresses of input sequence areas A through G in main storage.	28
AEND } through GENE }	End addresses of input sequence areas A through G in main storage.	28
AN8A10	Storage area for general registers 8, 9, and 10 during initialization of the equal routine.	12
ANINV	Base register sequence (8, 7) when initializing equal routine for sequence opposite to first control field.	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
	<p>This table contains the next disk address for all input/output sequences at any point in phase 2. It also contains index values (pointers to the work area table) for the current XTENT in which any given sequence disk address has been computed. The third entry in each subdivision (each sequence is a subdivision) is the logical unit address pertaining to each current or next sequence disk address.</p> <p>During a merge within a pass, only one output sequence is referred to. At the end of a merge, the output portion of the table is shifted up so that the next disk address is the address of the block of the new output sequence.</p>	
ASAVE	Save area for register 9 in PUTA routine (7 way fixed only).	4
BBCCHH	Seek and search address for input and output.	6
BCOMP	256 complement of BPT.	2
BLOCKC	Current sequence block count.	4
BPT	Number of sort blocks per track.	2
BPT1	Number of sort blocks per track, from checkpoint record.	2
CCBPERT	CCB for printing the pass number on SYSLOG.	16
CF1LCT } through } CFCLCT }	Information for control fields 1 through 12, from checkpoint record (8 bytes each).	96
CHECKP	Address of checkpoint track at end of work area. This address is loaded in register 3 at the end of phase 2 to provide retrieval of the checkpoint record by phase 3.	4
CHHR	Current disk interleave address.	4
CKPCCB	CCB for reading and writing checkpoint record	16
CKPCHP } through } RWCKPT }	CCW's for checkpointing every pass.	24
CPZERO	Constant 00F6 for converting pass numbers to decimal.	2
EXCESS	Maximum number of tracks by which an upper limit can be exceeded, when computing an interleaved disk address. Its only use is in computing the next disk address when an upper limit is exceeded.	2
FOUR	Constant 04 used to set condition code to a positive value in equal routine.	2
GBLOCK	Number of blocks in a sequence from checkpoint record (variable-length records only).	4
HCOMP	Constant 00FFF600 (256 complement of 10); used to allow switching to another cylinder in routine for calculating interleave address.	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
IG7A8	Save area for registers 7 and 8 in equal routine.	8
IGDOEX	Constant 18; used in equal routine.	2
IGHOLD	Save area used in equal routine for the base register obtained from mainline compare instruction.	2
INDEXL	Pointer to the extent within which the current address resides; used to "jump" to the next extent in the work area when the upper limit is exceeded.	2
INPUTG	Block count of input sequence (variable-length records only).	4
INPUTS	Save area for starting address of input area. The address can fluctuate between the value contained in ABEGIN-DBEGIN and an adjustment factor equal to the maximum record length minus one (variable-length records only).	4
IPTCCB	CCB for all input sequences. Before a "get" (read) is issued, the CCB contains the logical unit address (LUB table pointer) pertaining to the block about to be read in and the location of the channel program.	16
LIMITS	<p>Table of logical unit addresses and upper and lower limits associated with logical and physical segments of the work area. Each subdivision within the table is a 12-byte entry. There is a minimum of two and a maximum of seven subdivisions within the table, depending upon the number of extents the user has allotted for the work area.</p> <p>As illustrated in Figure 53, each entry contains:</p> <ul style="list-style-type: none"> • Lower and upper limits of each physical extent (4 bytes each). • Logical unit address associated with each work area extent (2 bytes). • A hexadecimal factor (multiple of 12) used as a pointer or index value to each physical extent (2 bytes). 	84

2 bytes	2 bytes	4 bytes	4 bytes
0000	Logical Unit Address 1	Lower Limit	Upper Limit
000C	Unit Address 2	Lower Limit	Upper Limit
0018	Unit Address 3	Lower Limit	Upper Limit
0024	Unit Address 4	Lower Limit	Upper Limit
0030	Unit Address 5	Lower Limit	Upper Limit
003C	Unit Address 6	Lower Limit	Upper Limit
0048	Unit Address 7	Lower Limit	Upper Limit

84 byte table in hexadecimal

-----EXAMPLE-----

Index Value	Log Unit Address	C-H-H-R	C-H-H-R
0000	0101	4-0-1-0	9-0-5-0
000C	0103	6-0-0-0	A-0-7-0
0018	0103	A-0-7-0	14-0-8-0
0000	0000	0-0-0-0	0-0-0-0
0000	0000	0-0-0-0	0-0-0-0
0000	0000	0-0-0-0	0-0-0-0
0000	0000	0-0-0-0	0-0-0-0

54 Tracks
47 Tracks
101 Tracks

101 tracks in each half of work area

User's extents for work area

Symbolic Unit Address	CCHH Lower Limit	CCHH Upper Limit
SYS001	0-4-0-1	0-9-0-4
SYS003	0-6-0-0	0-21-0-0

54 Tracks
151 Tracks
205 Tracks
- 3 Tracks for checkpoint
202 Tracks available

Figure 53. Phase 2 Limits Table

- LMALSB Checkpoint record format: 22
Number of passes - 2 bytes
Number of sequences from phase 1 - 4 bytes
Number of blocks in last merge, pass 1 - 4 bytes
Number of blocks in last sequence, pass 1 - 4 bytes
Number of records processed in phase 1 - 4 bytes
Number of blocks in G - 4 bytes
- LMBLOK Counter for the number of output blocks written during a pass when $S \leq M^2$. This count (at the end of a pass) contains the number of blocks that will be in the last merge of the next pass. This count is moved to LMSTRG which in turn is used in calculating when the output interleave factor is to be reduced (if at all) during the next pass. 4
- LMSTRG Counter used to determine when (if at all) the output interleave technique is to be reduced during a phase 2 pass. At the start of a pass, it is initialized with a count equal to the number of blocks contained in the last merge of a pass; that is, the number of input blocks remaining when $S \leq M$. 4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
	Whenever $S \leq M^2$, a check is initiated to see if the output interleave technique has to be reduced. The reduction is made after the number of blocks written to a sequence is 1 greater than the number of blocks contained in the last merge (M or less input sequences).	
LOGPHY	Table of physical work-area limits obtained from the checkpoint record; used in LIMITS table. (From TABLEB in assignment phase.)	88
LSTRG1 } through } LSTRG6 }	A maximum of six 4-byte counters used in computing the interleave factors for sequences:	24 max.
	LSTRG1 - A through F (7-way fixed) LSTRG6	24
	LSTRG1 - A through C (4-way fixed) LSTRG3	12
	LSTRG1 - A through E (6-way variable) LSTRG5	20
	LSTRG1 - A and B (3-way variable) LSTRG2	8
	These counters are used when it is determined that the interleave factors are to be reduced.	
	<u>Note:</u> The input interleave factor for the last sequence of a pass is not reduced. In a 7-way merge, for example, the input interleave factor is not reduced for sequence G, and a counter is not required.	
M	Phase 2 order of merge.	2
MAXFAC	A 2-byte constant which is used in conjunction with the interleaving of output sequences during a given pass. At the beginning of each pass, this constant is initialized with the number of merges which must be performed before any attempt is made to reduce the factors used in calculating output interleaved disk addresses. At the beginning of each merge within a pass, this constant is decremented by one. When the count reaches zero, the upcoming merge is notified that a check is to be undertaken for reducing the interleave technique associated with each output sequence. The count in MAXFAC cannot reach zero until S (number of input sequences) reaches a level that is equal to less than the order of merge squared ($S \leq M^2$).	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
MERGEL	The number of blocks contained in the last (input) sequence of a pass. It is moved to counters LSTRG1 - LSTRG6 which in turn are used in computing when the input interleave factor is to be reduced, if necessary. Because the number of blocks in the last merge of a pass becomes the number of blocks in the last sequence of the next pass, MERGEL does not have to be computed.	4
MOV1	Constant 0001; used during initialization of the move routine for fixed-length records.	2
MOV6	Constant 0006 used as the instruction length multiplier during initialization of the move routine for fixed-length record.	2
MOVIND	Save area for registers 8, 9, and 10 during initialization of the move routine.	12
MOVLN6	Input record length (L1) obtained from the checkpoint record (SORTRL); used during initialization of the move routine.	4
MOVMSK	Constant 255; used during initialization of the move routine.	4
NOPASS	Initially contains the number of passes to be performed by phases 2 and 3. This counter is decremented at the start of every pass. When the counter reaches zero, phase 3 will be called in. Example: NOPASS = 2-start of 1st pass, = 1-start of 2nd pass, = 0-fetch phase 3	2
NQUOT	The quotient of M/BPT. It is used in computing the head or track number (the value for the second H of CHHR) of interleaved disk addresses (also see NRMDR and example).	7 max
NRMDR	Remainders obtained when dividing the order of merge (M), order of merge minus one (M-1), etc., by the number of sort blocks per disk track. Used for computing the record numbers (R value in BBCCHHR) of each interleaved disk address associated with a sequence, both on input and output. Example:	7 max

<u>NQUOT</u>	<u>NRMDR</u>	
1	0	(M/BPT)
0	3	(M-1/BPT)
0	2	(M-2/BPT)
0	1	(M-3/BPT)

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
NSR	Output sequence counter used to initialize the input sequence counter (SR) at the beginning of each pass. NSR is then initialized with the number of output sequences which will be created during the upcoming pass (this value is derived by dividing the number of input sequences by the order of merge and rounding high).	4
OBEGIN	Starting address of output area in main storage.	4
OCCB	CCB for writing merged output on disk or tape. Functions in the same manner as IPTCCB.	16
OMERGE	Equated with PH2IOM; used for end-of-merge indicator	2
ONE	Constant 0001; used for incrementing sequence and block counters.	2
OPTWKA	Indicator for sort output in work area; used to test for an extra copy pass if the final output is to be in the work area.	2
ORADDP	Temporary storage area for output address and unit.	12
ORADDR	Output disk address table for sequences A through G. Same format as input, plus 8 bytes for shifting. See phase 2 constant ARADDR for table illustration.	84 max
OUTEND	End address of output area in main storage.	4
OUTPTG	Block count of output sequence.	4
OUTRLI	Used to extract the length of the current record (which may be a split record) about to be moved to the output area or to an input overflow area.	4
PASSNO	Pass number (in decimal) from key portion of checkpoint record; used for restart.	2
PH2IOM	Phase 2 order of merge.	2
PH2PASS	See PH42PAS.	17
PH42PAS	Message: "PHASE 2, PASS 00". 00 will be replaced by the appropriate pass number before message is printed.	17
PH3CST	Address of IPTCCB; used to initialize channel program (RWCKPT) to write constants for phase 3 on checkpoint track.	4
POINTI	Pointers (contained in checkpoint record) to the logical halves of the work area; used to initialize POINTL.	4
POINTL	Two hexadecimal pointers (plus two bytes for shifting) which indicate the starting points of the two halves of the work area. At the start of every pass, after the initial input addresses have been computed, these pointers are reversed (shifted) and the initial output addresses are computed.	6

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
PPCCB	CCB for writing PASS-PASS routine on the checkpoint track.	16
PPCHPG } through } RDWTPP }	CCW for reading and writing pass-pass routine on checkpoint track.	24
PRTPH2	CCW for printing messages on SYSLOG	8
QUOT	Used in computing the next head or track number (H value of CHHR) in compute interleave address routine. Similar in function to RMDR.	2
RDCHPG- } RDCCW }	CCW's for reading interleaved disk blocks into storage.	32
REC256	Constant 256; used for moving a record or a split record in the variable-length move routine.	2
RLCOND	Optional routine indicators.	2
RLINDE	Save area for registers 4 through 10 in relocater routine.	28
RLINVR } RLINVS }	Constant 00C0; used to initialize registers in relocater routine.	2 each
RLISA	Address of first byte of main storage following the relocated routines and available for the I/O area.	4
RLNG1	Original starting and ending addresses of the optional routines.	8
RLNG2	Modified starting and ending address of required optional routines.	8
RLREC2	Constant FFFFFFFD; used to round optional routine lengths to full word boundary.	4
RLSAVE	Save area for registers 5, 6, and 7 in relocater routine.	12
RLSVED	Area used in conjunction with RLSAVE to store the contents of registers in the relocater routine.	4
RMDR	2-byte field initialized each time an input/output interleave address is to be computed. When an interleaved disk address is computed, this field is added to the R value of the last address associated with an input/output sequence. RMDR is used only in computing the record number of the next disk address.	2
RSTCCB	CCB for checkpointing every pass.	16
SAVE45	Save area for general registers 4 and 5 (variable-length records only).	8
SAVPUT	Current address of the main storage output area (variable-length records only).	4
SFREG	Save area for register 15 in routine that computes when the input interleave factor is to be reduced.	4
SORTL	Sort block size obtained from the checkpoint record.	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
SORTRL	L1, Input record length - 2 bytes L2 - 2 bytes (not used by sort program) L3, Output record length - 2 bytes L4, Minimum record length - 2 bytes Output block size - 2 bytes Phase 3 or 4 exit indicator - 1 byte Record type and output rewind options - 1 byte Key length of user record - 1 byte Options (VERIFY, BYPASS, ADDR0UT, RESTART) - 1 byte	14
SR	The number of input sequences within a pass. During a pass, SR is decremented by the order of merge (M) at the start of every merge. At the start of every pass, SR is initialized with the number of output sequences created during the previous pass.	4
SYMADR	Logical unit (LUB) address; used to compute an interleaved disk address. Its function is similar to that of INDEXL whenever an upper limit is exceeded; i.e., the logical unit address of the next extent is extracted from the LIMITS table and placed in SYMADR which, in turn, is placed in the address entry, ARADDR through GRADDR or ORADDR, depending upon which sequence address has "jumped" to the new extent.	2
TEN	Constant FF00A00; used to determine the new input disk address when a logical upper limit has been exceeded.	4
WTCHPG - } WTCCW }	CCW's for writing the merged output onto disk.	32

PHASE 3 CONSTANTS

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
ABEGIN } through } GBEGIN }	Starting addresses of input sequence areas A through G in main storage.	28
AEND } through } GENE }	End addresses of input sequence areas A through G in main storage.	28
AN8A10	Storage area for general registers 8, 9, and 10 during initialization of the equal routine.	12
ANINV	Base register sequence (8, 7) when initializing equal routine for sequence opposite to first control field.	4
ANISEQ	Temporary storage for control field sequence. Used to test for control field sequence variation when initializing equal routine.	2
ANNML	Base register sequence (7, 8) when initializing equal routine for sequence same as first control field.	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
BPTO	Number of blocks per track in phase 3 output.	2
CANCEL	Upper case compare constant for "CANCEL" reply.	6
CANSEL	Lower case compare constant for "cancel" reply.	6
CCBCKP	CCB for reading checkpoint record.	16
CCBEOJ	CCB for printing end-of-job message on SYSLOG.	16
CF1LCT	Checkpoint record data for control fields 1 through 12 (8 bytes each).	96
CHECKP	Address of the checkpoint track obtained from general register 3 at the start of phase 3, and used for: <ul style="list-style-type: none"> • reading the checkpoint record (constants passed from phase 2 to phase 3) into main storage. • checkpointing the program for open, close, and FEOV conditions; in conjunction with this function, CHECKP is used to read in the label linkage routine (LLR) and the exit 31 routine. 	4
CHHR	Current disk interleave address.	4
CKPTRD	Data from key portion of checkpoint record: <ul style="list-style-type: none"> • Pass number (in decimal) for restart - 2 bytes • Number of passes - 2 bytes • Number of sequences from phase 1 - 4 bytes • Number of blocks in last merge (pass 1 of phase 2) - 4 bytes • Number of blocks in last sequence (pass 1 of phase 2) - 4 bytes 	16
COUNTR	Contains a decreasing record count during phase 3. The initial count is the number of records processed in phase 1. Every time a record is moved to the output area, this count is decremented by one. <u>Note:</u> Applicable only to 7-way fixed and 6-way variable.	4
CTDLDL	Output data length for a disk record.	2
DECMAL	Double word used in converting the number of records sorted into decimal format for printing end-of-job messages.	8
EIGHT	Constant 8; used in adjusting to the starting location of the output area in storage. It is used to "jump" to the location where the first record is to be moved to each output block in storage.	2
EOJPRT	CCW's for printing end-of-job messages on SYSLOG.	16
EOJPRT2	Location of CCW's for printing end-of-job messages.	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
EOV	End-of-volume indicator (V-character) for tape output only.	1
EX3231	Exits 32 and 31 indicators.	1
EXCESS	Maximum number of tracks by which an upper limit can be exceeded, when computing an input interleaved disk address. Its only use is in computing the next disk address when an upper limit is exceeded.	2
FFFF0	Constant used to invert control field digits in packed-decimal reversion routine.	16
FLTCP2	Constant used to invert control field digits in the floating-point reversion routine.	8
FORMAT	Format and number of control fields, obtained from the checkpoint record.	2
FOUR	Constant 04; used to set condition code to a positive value in equal routine.	2
HCOMP	Constant 00FFF600 (256 complement of 10); used to allow switching to another cylinder in routine for calculating interleave address.	2
IG7A8	Save area for registers 7 and 8 in equal routine.	8
IGDOEX	Constant 18; used in equal routine.	2
IGHOLD	Save area used in equal routine for the base register obtained from mainline compare instruction.	2
IGNOR	Lower case compare constant for "ignore" reply.	6
IGNORE	Upper case compare constant for "IGNORE" reply.	6
INDEXL	Pointer to the extent within which the current address resides; used to "jump" to the next extent in the work area when the upper limit is exceeded.	2
INVALID	CCW for printing message: "7DC2A INVALID RESPONSE" (see Appendix D).	16
IOLAB	Type of output and type of output labels. Byte 1: n1 - tape output 61 - nonstandard or unlabeled 21 - nonstandard 41 - unlabeled Byte 2: 01 - 1052 is assigned to SYSLOG.	2
IPTCCB	CCB for all input sequences. Before a "get" (read) is issued, the CCB contains the logical unit address (LUB table pointer) pertaining to the block about to be read in and the location of the channel program.	16

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
KEY	Key length of user record for disk output.	1
LIMITS	Table of logical unit addresses and upper and lower limits associated with logical and physical segments of the work area. Each subdivision within the table is a 12-byte entry. There is a minimum of two and a maximum of seven subdivisions within the table, depending upon the number of extents the user has allotted for the work area.	84

Each entry contains:

- Lower and upper limits of each physical extent (4 bytes each).
- Logical unit address associated with each work area extent (2 bytes).
- A hexadecimal factor (multiple of 12) used as a pointer or index value to each physical extent (2 bytes).

2 bytes	2 bytes	4 bytes	4 bytes
0000	Logical Unit Address 1	Lower Limit	Upper Limit
000C	Logical Unit Address 2	Lower Limit	Upper Limit
0018	Logical Unit Address 3	Lower Limit	Upper Limit
0024	Logical Unit Address 4	Lower Limit	Upper Limit
0030	Logical Unit Address 5	Lower Limit	Upper Limit
003C	Logical Unit Address 6	Lower Limit	Upper Limit
0048	Logical Unit Address 7	Lower Limit	Upper Limit

84-byte table in hexadecimal

EXAMPLE

Index Value	Log Unit Address	C-H-H-R	C-H-H-R	
0000	0101	4-0-1-0	9-0-5-0	54 tracks
000C	0103	6-0-0-0	A-0-7-0	47 tracks
0018	0103	A-0-7-0	14-0-8-0	101 tracks
0000	0000	0-0-0-0	0-0-0-0	
0000	0000	0-0-0-0	0-0-0-0	
0000	0000	0-0-0-0	0-0-0-0	
0000	0000	0-0-0-0	0-0-0-0	

101 tracks in each half of work area

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
LIMITO	Upper limit of the current disk output extent. If the upper limit is exceeded, an OPEN must be issued to retrieve the next set of extents. For tape output, the constant is used in compiling the block count to be inserted in the trailer label. (Standard labels only.)	4
LINKRT	80-byte area reserved for IOCS to read the standard tape label block into main storage when processing standard tape file labels.	80
LOGADR	Logic module address in tape DTF.	4
LSTRG1 } through } LSTRG6 }	A maximum of six 4-byte counters used in computing the interleave factors for sequences:	24 max.
	LSTRG1 - A through F (7-way fixed) LSTRG6	24
	LSTRG1 - A through C (4-way fixed) LSTRG3	12
	LSTRG1 - A through E (6-way variable) LSTRG5	20
	LSTRG1 - A and B (3-way variable) LSTRG2 (See note following LSTRG1-LSTRG6 in the phase 2 constants.)	8
MESE0J	Message: "7DC5I END OF SORT".	17
MESE0J2	Location of end-of-job message (MESE0J).	4
MNLADR	Address of CCW's for reading and writing the label-linkage routine.	4
MNLCCB	CCB for checkpointing mainline prior to label processing.	16
MOV1	Constant 0001; used during initialization of the move routine for fixed-length records.	2
MOV6	Constant 0006; used as the instruction length multiplier during initialization of the move routine for fixed-length records.	2
MODADDR	Logic module address in disk DTF.	4
MOVIND	Save area for registers 8, 9, and 10 during initialization of the move routine for fixed-length records.	12
MOVLNG	Input record length (L1) obtained from the checkpoint record (SORTRL); used during initialization of the move routine for fixed-length records.	4
MOVMSK	Constant 255; used during initialization of the move routine for fixed-length records.	4
MOVRES	Save area for the calculated number of moves; used during initialization of the move routine for fixed-length records.	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
NOPH3C	Main storage address and the byte count for reading in constants being passed from phase 2 to phase 3.	7
OBEGIN	Starting address of output area in main storage.	4
OCCB	CCB for writing merged output on disk or tape. Functions in the same manner as IPTCCB.	16
ONE	Constant 0001; used for incrementing sequence and block counters.	2
OPEN	Indicator for the label-linkage routine that an "OPEN" is to be issued to the output file.	1
ORADDR	Current 2311 address of the output file on disk. At the start of phase 3, it contains a value which will force the initial "OPEN" of the output file before the first output block is written onto the disk output area.	4
OUTBKL	Output block length (for fixed-length records), or maximum output block length (for variable-length records).	2
OUTEND	End address of output area in main storage. <u>Note:</u> This address may be replaced by a lower value after an output block has been written on disk when processing VLR <u>only</u> . OUTEND will contain the original value (OUTEND1) at least until the first block has been built and the amount of track capacity that will remain has been calculated. If this amount is less than the maximum output block size, the output area is truncated to equal the track capacity remaining, and the end address of the truncated output area is inserted in OUTEND for testing the next record.	4
OUTEND1	For processing variable-length records only, the original end address of the main storage output area (OUTEND) is stored in OUTEND1, which can then be used to restore the truncated output area to its original size.	4
OUTRDL	Output record lengths: First 2 bytes - output record length (fixed-length records) or maximum output record length (variable-length records).	2
PDIGITS	Save area for shifted control data in packed-decimal reconversion routine.	16
PH34EX	Name of phase 3 user routine.	4
PH3IOM	Phase 3 order of merge (the number of sequences remaining at the beginning of phase 3).	2
PH3REG	Same as first CCW in RDCHPG.	4
PKSGN2	Temporary storage area for sign bits in the packed-decimal reconversion routine.	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
QUOT	Used in computing the next head or track number (H value of CHHR) in compute interleave address routine. Similar in function to RMDR.	2
RDCHPG -} RDCCW	CCW's for reading interleaved disk blocks into storage.	32
RDCKPT -} CKPCCW	CCW's for reading in the checkpoint record and the record containing applicable constants. Used during initialization of phase 3.	32
RECMES	Message: "7DC4I RECORDS PROCESSED 0000000".	31
RECMES2	Address of message contained in RECMES	4
RECPH1	Number of records processed in phase 1, obtained from checkpoint record.	2
REPLY	Operator's response (in characters) to messages, if SYSLOG is a 1052.	6
REPLYBAD	Message: "7DC2A INVALID RESPONSE" (see Appendix D).	
REWIND	Rewind options for tape output: 10 - RWD (rewind) 04 - RWD UNLD (rewind unload) 01 - NORWD (no rewind)	1
RLADRU	Starting address of original optional routines in phase 3 relocater.	4
RLCOND	Area for optional routine indicators.	2
RLEMG	Length of compare set.	4
RLINDE	Save area for registers 4 through 10 in relocater routine.	28
RLINVR } RLINVS }	Constant 00C0; used to initialize registers in relocater routine.	2 each 2
RLIOUT	Overlays BLOCKC from phase 2.	4
RLISA	Address of first byte of main storage following the relocated routines and available for the I/O areas.	4
RLKOTO	Constant 16; used to test for last selector.	4
RLNG1	Table of original optional routine lengths.	24
RLNG2	Table of modified optional routine lengths.	24
RLREC2	Constant FFFFFFFD; used to round optional routine lengths to full word boundary.	4
RLRSVE	Save area for zone bits in relocater move routine.	2
RLSAVE	Save area for registers 5, 6, and 7 in relocater routine.	12
RLSVED	Area used in conjunction with RLSAVE to store the content of registers in the relocater routine.	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
RMDR	2-byte field initialized each time an input interleaved address is to be computed. When an interleaved disk address is computed, this field is added to the R value of the last address associated with an input sequence. RMDR is used only in computing the record number of the next disk address.	2
SAVE10	Save area for general register 10 in the record count routine (7-way fixed only).	4
SEQCCB	CCB for printing sequence-error message and for reading reply on SYSLOG.	16
SEQCCW	CCW's for printing sequence error message and for reading reply on SYSLOG.	16
SEQMES	Message: "7DC2D SEQ. ERROR" (see Appendix D).	16
SLINK	Save area for the contents of the linkage register (14) before entering user exit 32.	4
SORTL	Sort block size, obtained from the checkpoint record.	2
SORTRL	L1, input record length.	2
SYMADR	Logical unit (LUB) address; used to compute an interleaved disk address. Its function is similar to that of INDEXL whenever an upper limit is exceeded; i.e., the logical unit address of the next extent is extracted from the LIMITS table and placed in SYMADR which, in turn, is placed in the address entry, ARADDR through GRADDR or ORADDR, depending upon which sequence address has "jumped" to the new extent.	2
TADCON	Address of CCW for writing merged output onto tape or disk.	4
TC3633	Constant 3633, the 2311 track capacity in bytes for writing disk output. The track capacity is actually 3625, but the same result is achieved in phase 3 by increasing the value eight bytes.	2
TEN	Constant FF00A00; used to determine the new input disk address when a logical upper limit has been exceeded.	4
TKLEFT	Counter for computing the remaining area on a 2311 track. Every time a record is written onto the disk file, this counter is adjusted to indicate how many bytes remain on a track.	2
USADDR	Main storage address of the user programming (if any) in phase 3. It is used to link the sort program to exit 31 and exit 32.	4
USERINFO	<ul style="list-style-type: none"> • Output record length (L3) - 2 bytes • Input record length (L1 for fixed or variable-length records, or control field length plus 10 for ADDRROUT records) - 2 bytes 	8

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>								
	<ul style="list-style-type: none"> • Label type, ADDRROUT option, format and record type - 1 byte • Control field 1 length minus 1 - 1 byte • Control field 1 displacement within record - 2 bytes 									
VBPRC	<table border="1" style="margin-left: 40px;"> <tr> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> <td style="text-align: center;">-</td> <td style="text-align: center;">V</td> <td style="text-align: center;">B</td> <td style="text-align: center;">P</td> <td style="text-align: center;">R</td> <td style="text-align: center;">C</td> </tr> </table>	-	-	-	V	B	P	R	C	1
-	-	-	V	B	P	R	C			
	<p>V - VERIFY option (disk only).</p> <p>B - BYPASS (tape input for sort, tape input or output for merge).</p> <p>P - PRINT option.</p> <p>R - ADDRROUT option (disk input only).</p> <p>C - RESTART option (sort only).</p>									
WRMAIN	For open, FEOV, and close conditions, this channel program will checkpoint phase 3, read in the label-linkage routine, checkpoint the label-linkage routine after the condition has been satisfied, and restore phase 3 for further processing (if warranted).	32								
WTCHPG } through } WTCCW }	CCW's for writing the merged output onto disk or tape (only WTCCW is used for tape output). Also used to verify disk output.	56								
X31IND	Open, close, FEOV, or sequence-error indicator	1								
ZONZN2	Temporary storage for sign bit in the zoned-decimal reconvension routine.	2								

PHASE 4 CONSTANTS

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
ABEGIN } through } DBEGIN }	Starting addresses of input areas in main storage for files A, B, C, and D, respectively.	16
ACCB } through } DCCB }	Command Control Blocks for input files A through D.	64
ADD2	Disk address of the checkpoint.	7
ADDRINP *	Address of FILE A CCB.	4
ADDRROUT *	Address of output CCB.	4

*Appears in DSORT401 only.

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
AEND } through DEND }	Ending addresses of input areas in main storage for files A, B, C, and D, respectively.	16
AN8A10	Storage area for general registers 8, 9, and 10 in initialization of the equal routine.	2
ANINV	Base register sequence (13, 12) when initializing equal routine for sequence opposite to first control field.	4
ANISEQ	Temporary storage for control field sequence. Used to test for control field sequence variation in equal routine initialization.	2
ANNML	Base register sequence (12, 13) when initializing equal routine for sequence same as first control field.	4
ARADDR } through DRADDR }	Current disk address and extent upper limit for each file (A, B, C, and D).	32
BASE11 **	Starting address of phase 4 overlays.	4
BBCCHH	Part of disk seek address. When combined with CHHR, forms the disk seek address for the 2841 control unit.	3
BCOMPO	256 complement of the maximum number of output blocks per track (BPTO). Used only for disk output, fixed length records.	2
BLKINP	Input block length of fixed blocks or maximum block size of variable blocks. (See INPBKL.)	2
BLX537	Constant 537 used in computing the area of a 2311 track that will be used in writing variable length block (record) onto a track. It applies only to disk output.	2
BPTO **	Number of blocks per track in phase 4 output.	2
BYPCCB *	Command control block for printing bypass message.	16
BYPCCW *	CCW for printing bypass message.	8
BYPMES *	Message: "7DD4I PHASE 4 UNREADABLE BLOCKS BYPASSED 00000". (The zeros are replaced by the number of blocks bypassed.)	46
CCB1	CCB for writing initialized open/close routine on the second track of the disk work area.	16
CCBEOJ	CCB for printing messages on SYSLOG.	16
CCBTAB } through SEARCH+31 }	For reading in the assignment phase table.	56
CCOUNT	Count field for current input block (CCHHRKLDL). Used for disk input only.	8

 **Appears in both both DSORT401 and DSORT402.

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
CCW1 } to CCW2 }	CCW chain for writing initialized open/close routine on disk.	32
CF1LCT **	Checkpoint record data for control fields 1 through 12 (8 bytes each).	96
CHHR	Computed current 2311 disk address.	4
CHKCCB	CCB for checkpoint CCW (CHKCCW).	16
CHKCCW	CCW's for seek, search, read, and write checkpoint for open or close conditions.	32
CHKPT	Constant: /// CHKPT //	12
COUNT **	In DSORT401, count for the format of the checkpoint record and area for labels processing. In DSORT402, count of next record.	80 8
DECIMAL *	Double word used for converting the number of records merged to decimal form when printing bypass and end-of-job messages.	8
DISK1 } * DISK2 }	Disk addresses used in seek and search CCW's.	4 4
DTFADDR *	Address of DTF for OPEN and CLOSE macros.	4
EOFDK *	Address of CLOSE macro for input file at end-of-file time for disk; equated to EOFTP.	4
EOFTP *	Address of macro to close input file at end-of-file time for tape.	4
EOJPRT *	CCW's for printing end-of-merge messages.	16
ERRCT	Count of unreadable blocks bypassed.	2
FILETYPE *	File information for user during exit 41.	4
FIXDRL	Input record length of fixed length records or maximum length of variable records. (See SORTRL.)	2
FORMAT **	Format and number of control fields obtained from checkpoint record.	2
HCOMP	256 complement of ten. Used for computing next output disk address for variable-length records.	4
IGDOEX	Constant 18 used in equal routine.	2
IGHOLD	Save area used in equal routine to hold register number of the base register obtained from mainline compare instruction.	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>																					
INPBKL	Input block size (maximum input block size for variable-length records) from checkpoint record.	4																					
INOUT	Equivalent to assignment phase INPUTMRG through INPUTMRG+4.	5																					
KEY **	<u>First byte:</u> Key length <u>Second byte:</u> Bit 1 - unused. Bit 2 - NOTPMK before first tape output record Bit 4 - VERIFY option; disk output only Bit 4 - BYPASS option; tape input only	2																					
LABEL	Label type indicator	1																					
LENGTH *	Length of initialized open/close routine written on disk.	4																					
LIMITO	Upper limit of the current disk output extent. Equals ORADDR+4.	4																					
LOGUNIT *	Current logical unit address for work area.	4																					
MASK	Constant FFFFFFFC; used to create a full word boundary for the output area.	4																					
MES1 *	Upper case compare constant for "IGNORE" reply.	6																					
MES2 *	Upper case compare constant for "CANCEL" reply.	6																					
MES11 *	Lower case compare constant for "ignore" reply.	6																					
MES22 *	Lower case compare constant for "cancel" reply.	6																					
MESE0J *	Second part of end-of-job message in DSORT401: "7DD6I END OF MERGE". Also see RECMES.	18																					
MRGVOL **	Number of volumes and files for merge from assignment phase checkpoint record: <table border="1" style="margin-left: 40px;"> <thead> <tr> <th>FILE</th> <th>A</th> <th>B</th> <th>C</th> <th>D</th> <th colspan="5"></th> </tr> </thead> <tbody> <tr> <td>Number of volumes per file</td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px;"></td> <td style="width: 20px; height: 20px; text-align: center;">-</td> <td style="width: 20px; height: 20px; text-align: center;">-</td> <td style="width: 20px; height: 20px; text-align: center;">-</td> <td style="width: 20px; height: 20px; text-align: center;">-</td> <td style="width: 20px; height: 20px; text-align: center;">-</td> <td style="width: 20px; height: 20px;"></td> </tr> </tbody> </table> <p style="margin-left: 100px;">↑ Total number of files (MRGVOL+9)</p>	FILE	A	B	C	D						Number of volumes per file					-	-	-	-	-		10
FILE	A	B	C	D																			
Number of volumes per file					-	-	-	-	-														
NOOPR	4700; used to change branch instructions to no-ops during initialization for fixed-length records on input.	2																					
OCCB	Command control block for output file.	16																					
OM **	Phase 4 order of merge, from checkpoint record (see assignment phase label FILES).	2																					
OMERGE *	Phase 4 order of merge.	4																					
ONE	Constant 1; used in calculations.	1																					
ORADDR	Current disk address of the output file.	4																					

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
OUTBKL **	Output block size (maximum output block size for variable-length records).	2
OUTEND	End address of output area in main storage.	4
	<u>Note:</u> This address may be replaced by a lower value after an output block has been written on disk when processing VLR <u>only</u> .	
	OUTEND will contain the original value (OUTEND1) at least until the first block has been built and the amount of track capacity that will remain after it is written has been calculated. If this amount is less than the maximum output block size, the output area is truncated to equal the track capacity remaining, and the end address of the truncated output area is inserted in OUTEND for testing the next record.	
OUTEND1	For processing variable-length records only, the original end address of the main storage output area (OUTEND) is stored in OUTEND1, which can then be used to restore the truncated output area to its original size.	4
OUTLGN	Output record length of fixed-length records or actual length of variable records. (See OUTRDL.)	2
OUTOPT **	Record type and output label option; equated to TYPREC:	1
	Bit 0 - Variable-length records Bit 1 - Fixed-length records Bit 2 - OPEN RWD Bit 3 - OPEN NORWD Bit 4 - not used Bit 5 - CLOSE UNLOAD Bit 6 - CLOSE RWD Bit 7 - CLOSE NORWD	
OUTRDL **	Output record lengths:	4
	First 2 bytes - output record length (fixed-length records) or maximum output record length (variable-length records).	
	Second 2 bytes - minimum variable-length record.	
OUTRLI	Length of record to be moved to output area.	4
OVF	Characters O, V, and F; used to indicate open, end of volume, and end of file, respectively.	3
PH2IOM	Merge status indicator.	2
PH2MOM	Equated to PH2IOM.	2
PH34EX **	Phase 4 user exit number	2

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
PH4NAME **	Symbolic name for phase 4 user routine	8
PKSGN2	Temporary storage area for sign bits in the packed decimal reconversion routine	2
R1CTWD } and R2CTWD }	Disk address of the assignment phase checkpoint record.	4
RCOUNT	Number of records processed (merged). Does not include user insertions or deletions.	4
RDCCW	For reading records from a file into the input area (tape input only).	8
RDCHPG	Same as RDCCW, except for disk input.	56
READTP	Address of RTCCW; used to initialize input channel programs ACCB through DCCB to read tape input.	3
REC256	Constant 256; used in calculations for moving records to the output area.	4
RECMES *	First part of end-of-job message in DSORT401: "7DD5I RECORDS PROCESSED 0000000". (The zeros are replaced by the number of records processed.) Also see MESE0J.	33
RLADRU	Starting address of original optional routines in phase 4 relocater.	4
RLCOND	Indicator for relocatable routines.	1
RLEMG	Length of compare sets.	4
RLINDE	Save area for registers 4 through 10 in relocater routine.	28
RLINVR } RLINVS }	Constant 00C0, used to initialize registers in relocater routine.	2 each
RLISA	Address of first byte of main storage following the relocated routines and available for the I/O area.	4
RLKOTO	Constant 16; used to test for last selector.	4
RLNG1	Table of original optional routine lengths.	24
RLNG2	Table of modified optional routine lengths.	24
RLREC2	Constant FFFFFFFD; used to round optional routine lengths to full word boundary.	4
RLRSVE	Save area for zone bits in move routine.	2
RLSAVE	Save area for registers 5, 6, and 7 in relocater routine.	12
RLSVED	Area used in conjunction with RLSAVE to store the content of registers in the relocater routine.	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
RTCCW	CCW for reading tape.	8
SAVE45	Storage location for saving the contents of registers 4 and 5; used in moving records to the output area.	8
SAVE0	Storage location for saving the contents of register MREG when branching to open/close routine.	4
SAVEREG *	Save area for general registers 10 through 15.	24
SEEK *	Seek and search CCW's for reading assignment phase checkpoint record.	32
SEQCCB1 through SEQCCB3 }	CCB's for printing sequence error messages.	48
SEQCCW1 through SEQCCW3 }	CCW's for printing sequence error messages.	24
SEQMES1 through SEQMES3 }	Sequence error messages 7DD2A and 7DD2D; see Appendix D.	51
SLINK	Storage location for saving the contents of the link register.	4
SORTRL **	L1, Input record length - 2 bytes L2 - 2 bytes (not used by sort program)	4
TABCCB *	CCB for reading the assignment phase table.	16
TABLEB **	Disk work area limits table from checkpoint record. TABLEB is read in with the checkpoint but is not used in phase 4.	88
TBKLAB **	Type of blocking (input) and input file options: Bit 0 - Variable blocking Bit 1 - Fixed blocking Bit 2 - OPEN RWD Bit 3 - OPEN NORWD Bit 4 - not used Bit 5 - CLOSE UNLOAD Bit 6 - CLOSE RWD Bit 7 - CLOSE NORWD	1
TC3656	Constant 3633; the capacity, in bytes, of the 2311 track. For disk output only.	2
TKLEFT	Counter for number of bytes remaining on a track. Used for computing the available area on a 2311 track after record has been written on disk. For variable-length output only.	2
TPOM *	Number of tapes in input.	4
TRACK1 ** TRACK2 }	Addresses of first and second tracks of work area.	4

<u>Label</u>	<u>Contents</u>	<u>Bytes</u>
TRK1 } TRK2 }	Addresses for writing initialized open/close routine on disk.	4
TYPREC **	Type of record (fixed or variable) and output label options (type from RECTYP, options from INOUT+1 in assignment phase). Second byte: Bit 0 - Variable-length records Bit 1 - Fixed-length records Bit 2 - OPEN RWD Bit 3 - OPEN NORWD Bit 4 - not used Bit 5 - CLOSE UNLOAD Bit 6 - CLOSE RWD Bit 7 - CLOSE NORWD	2
USADDR **	Starting address of user routine in phase 4.	4
USERAD **	Address of phase 4 user routines (same as USADDR).	4
USINF1	Input block length for user.	4
USINF2	First 2 bytes - Output record length (L3) for user. Second 2 bytes - Input record length (L1) for user.	4
USINF3	First byte - record type, format, and label type for user. Second byte - First control field information (CFILCT) for user. Third and fourth bytes - unused.	4
VRFCCW	CCW for verification of records written on disk.	8
WLRCCB	Command control block for wrong-length record message.	16
WLRCCW	CCW for printing wrong-length-record messages.	8
WLRMES	Wrong-length-record message: '7DD1I WLR FILE X' (The X is replaced by the file identification.) See Appendix D.	16
WRITETP	Used to modify WTCCW for tape.	3
WTCHPG } through } WTCCW+24 }	For writing merged records from the output area onto disk or tape (modified by WTPCCW for tape output).	48
WTPCCW	CCW used to modify WTCCW program for tape output.	8
XFILE **	Current file indicator for input files A through D.	1
XTIME	Indicator for conversion or reversion of data.	1
ZONZN2	Temporary storage for sign bit in the zoned-decimal reversion routine.	2

DEFINITIONS OF ABBREVIATIONS

BAL - Branch-and-link
BLI - Block length indicator
BPT - Sort blocks per track
CCB - Channel command block
CCW - Channel command word
CF - Control field
CHHR - Cylinder, head-head, record number
EOF - End of file
EOJ - End of job
EOV - End of volume
G - Number of records per sequence in phase 1
Hex - Hexadecimal
LLR - Label linkage routine
M - Order of merge
OM - Order of merge
RAF - Record address file
RLI - Record length indicator
RPT - Records per track
S - Number of input sequences remaining at the start of a merge within a pass
TA - Tracks available (assignment phase). Used in calculating disk work area
TC - Track capacity
TR - Tracks required (assignment phase). Used in calculating disk work area
VLR - Variable-length record
WLR - Wrong-length record

APPENDIX B: LABEL REFERENCES

ASSIGNMENT PHASE

<u>Label</u>	<u>Chart</u>
ABORT	AA
AGAIN	AQ
B1	AH
B2	AH
B118	AP
B119	AP
B126	AP
B135	AP
B136	AP
B170	AP
BEGIN1	AA
BEGIN	AP
BLK	AG
BLK2	AG
BLKHIP	AJ
BLKOK	AJ
BLMAX4	AH
BRCH	AR
BYPVAR	AL
C	AJ
CALC	AQ
CALCSW1	AK
CALCSW2	AL
CALNOP	AM
CALPH1B2	AK
CALRPT	AL
CALSWR2	AN
CALSW2V	AM
CBYPTK	AK
CBYPTKV	AM
CCDESC	AR
CCERR	AA
CCS0	AB
CCS2	AB
CF256	AG
CFB3MAX	AH
CFL3MAX	AH
CHKRECBK	AL
CKAMAX	AM
CKBI	AH
CKBMAX	AH
CKBPT	AK
CKBYPTKR	AN
CKDEFS	AE
CKFLD1	AD
CKFLD4	AD
CKFLDM	AD
CKHEAD	AR
CKKEY	AJ
CKLABELS	AJ
CKL3MAX	AH
CKL3MX	AJ
CKLMAX	AH
CKMERGE	AJ
CKPARA	AA

CKPRINT	AR
CKRECBK	AL
CKRECBKR	AN
CKRPT	AN
CKRPT1	AN
CKRWD	AJ
CKSIZ	AJ
CKTYPEIN	AJ
CKVOL	AJ
CLEAR	AB,AF
CLOBV	AH
CLUB	AP
COL16	AA
COMBPT	AK
COMBPTV	AM
COMP	AC,AD,AQ
COMPG	AL
COMPGR	AN
COMPGV	AM
COMPM	AD
COMPO	AD
COMPP	AE
CONOUT	AR
CORVAL	AG
CVB3MAX	AH
CVL3MAX	AH
DETLLOWER	AH
DETSMALL	AH
DIV8L	AM
DSORT	AA
DSORT002	AB
DSORT003	AC
DSORT004	AD
DSORT005	AF
DSORT006	AH
DSORT007	AJ
DSORT008	AK
DSORT009	AP
DSORT010	AR
E181	AP
E200	AP
ENDGF	AL
ENDGFR	AN
ENDSCN	AE
EOJ	AR
EOJ3RT	AS
EOJC1	AS
EOJCALC	AR
EOJCSW	AS
EOJZ	AR
ERR1	AA
ERR3	AA
ERR3A	AA
ERR4	AA
ERR5	AA
ERR6	AA
ERR64	AA
EXIT	AF
EXITB	AF
F2SW	AS
F4SW	AS
FETC10	AP
FETCH	AA
FETCH1	AS
FETCH2	AS

FETCH4	AC, AS
FETCH5	AE
FETCH7	AH
FETCH9	AL, AM
FETCHEND	AE, AH, AM
FETVAR	AS
FINAL	AP
FIND	AQ
FIRSTED	AG
FIXED	AP
GETEOJ	AG
GETL1	AJ
GETMES	AB
INCSWR	AN
INF	AR
INF1	AD
INFSCN	AD
INIT1	AF
INIT2	AF
INITA1	AF
INITST	AA
INITCSWS	AL
L1	AH
L1MULT	AJ
L3L1	AJ
LABEXCK	AJ
LENHI	AJ
LENLO	AJ
LOAD1	AA
LOAD2	AB
LOOP	AJ
M1	AR
METH1I	AF
MGINIT	AF
MINBLKI	AJ
MOD1	AD
MODSCN	AD
MOVECC	AB
MRGCKP	AF
MRGOPN	AF
MRGSCN	AC
MSG91	AS
MSG91A	AS
NOCALCS	AR
NORAF	AJ
NOVAR	AG
ONETWO	AQ
OPEN	AF
OPERROR	AA
OPFLE	AR
OPTSC1	AA
OPTSC2	AA
OPTSCN	AE
OPTSCS	AA
OTFSCN	AD
OUT	AQ
OUTFI	AR
OUTPT	AL
OVLPCF	AJ
P1SIZE	AG
P2SIZE	AG

P3SIZE	AG
P4NAM	AG
P4NAMS	AG
P4SIZ	AG
PHZ2	AL
PRINTE	AP
PRNTCARD	AR
PRTCALCS	AR
PUTCNSWR	AN
PUTCO2	AL
PUTCO2R	AN
PUTCON	AL
PUTCONSW	AL
PUTCOV	AM
RAFBI	AH
RAFRTN	AN
RC1	AC
RCDSCN	AC
RDCKPR	AS
RDRTN	AA
RE1	AR
READB	AA
READX	AF
READX1	AF
READXT	AF
REINIT	AL
REINIT1	AL
REINTI1R	AN
REINITR	AN
REDUCE	AK
REDUCER	AN
REDUCEV	AM
REP1	AP
REPB	AP
RESTOR	AP
RETRY	AL, AP
RT1	AG
RUNCODSW	AA
S188	AP
SAVCKP	AF
SCAN	AC
SEEKTM	AQ
SORTXT	AF
SPADE	AP
SRT1	AC
SRTSCN	AC
ST1-4	AF
STBO	AN
STDFV	AG
STEP	AK
STEP1	AK
STEP1B	AK
STEP1BR	AN
STEP1R	AN
STEP1V	AM
STEP3	AK
STEP3A	AK
STEP4	AK
STEP4R	AN
STORE	AD
STOREM	AD
STORP	AE
STVSOM	AP
SVORIG	AM
SW	AA
SW1	AA, AL

SW1R	AN
SW2V	AM
TABLEF	AQ
TAGOUT	AP
TEST	AB
TESTDEV	AJ
TESTOPERR	AE
TSER7	AJ
TSMERG	AR
TSREST	AS
TSTBLKCD	AA
TSTCALCS	AR
TSTEND	AB
TSTER5	AG
TSTLGH	AB
UPMOVE	AA
USEORIG	AK
USEORIGR	AN
USESTR	AG
VALIDATE	AJ
VARBI	AH
VAREND	AP
VARRTN	AM
VBI	AH
VLI	AH
VLESSG	AP
WTCKMG	AR
WTCKPS	AS

PHASE 1

<u>LABEL</u>	<u>CHART</u>
BLDADR	BK
CHKPT	BA
CHKUSER	BA
CHNCMP	BJ
CHNMVE	BK
CMPCHN	BF
COMPRES	BH
CYLINC	BA
DCLOSE	BM
DIBLRD	BD
DIHERE	BD
DINGET	BK
DINGUD	BK
DINJCY	BD
DINLES	BK
DINMVA	BD, BK
DSKINIT	BA
ENDINP	BM
ENDMAC	BN
EOFADDR	BN
EOFINIT	BA
EOVEOF	BN
EXIT	BF
EXITP1	BH
EXITSW	BA

FECH22	BB
FEOVBR	BN
FEOVRTN	BN
FET102	BA
FETCH	BB, BG, GH
FLMVBC	BC
FLMVGO	BC
FORMAT	BB
FORMGO	BB
FRMBPT	BB
FRMCYB	BB
FRMGWK	BB
FRMSTT	BB
FRMXTC	BB

GETXT	BM
-------	----

IBV001	BG
IBV021	BB
IBV022	BC
IBV023	BJ
IBV024	BH
IBVAA1	BC, BJ
IBVIF4	BF
IBVJC2	BF
IBVMC3	BG
IBVMH4	BG
IBVND4	BG
IBVNE4	BG
IBVPB2	BG
IBVVL	BG
INEXIT	BD, BL
INFINI	BD, BL
INIT32	BA
INITE11	BA
INMOVE	BD
INPTIZ	BC
INTAPE	BA
INTRED	BE
IZTAPE	BC

LBLCHK	BL
LBLINIT	BA
LOCARE	BC, BJ
LSTFLSW	BM
LSTFILSW	BN

MNLDSK	BM
MNLLINK	BM
MNLTAP	BM
MOPS1	BG

NOEOB	BK
NONSTD	BA
NXTFILE	BN
NXTREC	BK

ONEXIT	BD, BL
OPENA	BM
OPENT	BM
OUTIZ	BC, BJ
OV2INW	BH

P1INBG	BD, BK
P1LEV1	BF
PADRTN	BG

RDCNT	BK
REDOUT	BF
SAVADR	BA
SAVVOL	BN
STORM1	BF
STORM2	BF
TAGADR	BK
TAGCHK	BJ
TAGINT	BJ
TAGMNL	BK
TINBG	BE
USEREX11	BM
VARBLK	BK
VLI125	BC
VLINAA	BC
VLINAC	BC
VLRINZ	BC, BJ
WLR	BK

PHASE 2

<u>Label</u>	<u>Chart</u>
BSTR6	CM, CY
BYPAS1+4	CD, CR
BYWCKD	CA, CN
CALADR	CM, CY
CKPTOK	CC, CQ
COMPBA	CK, CW
COMPCA	CJ, CV
COMPGB	CJ, CV
COMPDA	CH, CU
COMPDB	CH, CU
COMPDC	CH, CU
COMPEA	CG, CT
COMPEB	CG, CT
COMPEC	CG, CT
COMPED	CG, CT
COMPFA	CF, CS
COMPFB	CF, CS
COMPFC	CF, CS
COMPFD	CF, CS
COMPFE	CF, CS
COMPGA	CE
COMPGB	CE
COMPGC	CE
COMPGD	CE
COMPGE	CE
COMPGF	CE
COMPIT	CA, CN
CPIOAS	CA, CN
CPOADR-8	CC
CPOADR-12	CQ
DIVAGN	CA, CN
ENDPAS	CB, CP
EXECPP	CC, CQ

FETCH3	CC, CQ
FILLA	CD, CR
FILLB	CD, CR
FILLC	CD, CR
FILLD	CD, CR
FILLE	CD, CR
FILLF	CD, CR
FILLG	CD
GETA	CD, CR
GETB	CD, CR
GETC	CD, CR
GETD	CD, CR
GETE	CD, CR
GETF	CD, CR
GETG	CD
ILEAVE-10	CC, CQ
INITOM	CC, CQ
INTPH2	CA, CN
IRMDR	CD, CR
IRMDR1	CD, CR
LM1	CD, CR
LM1234	CB, CP
LMO	CL, CX
LMOSW	CB, CP
LMTSOK	CM, CY
MMPP	CL, CX
MMPP1	CM, CY
MMPP2	CB, CP
MMPPS	CB, CP
MOV MVC	CL
NEWITL	CC, CQ
NEXTPASS	CC, CQ
NOCHG	CB, CP
NOTLAS	CC, CQ
NOWCKD	CA, CN
NXTAR	CK, CW
NXTBR	CK, CW
NXTCR	CJ, CV
NXTDR	CH, CU
NXTER	CG, CT
NXTFR	CF, CS
NXTGR	CE
OLEAVE	CC, CQ
ORMDR	CL, CX
ORMDR1	CL, CX
OSPLIT	CY
OUP TOK	CA, CN
OUTFUL	CL, CX
PUTA	CK, CW
PUTB	CK, CW
PUTC	CJ, CV
PUTD	CH, CU
PUTE	CG, CT
PUTF	CF, CS
PUTG	CE
RCHKPT	CA, CN
RDABCD	CD, CR
RDCPOK	CA, CN
REDUCEI	CB, CP
REDUCEO	CB, CP

SFOADR	CB,CP
SHTPTR	CC,CQ
SPLITA	CW
SPLITB	CW
SPLITC	CV
SPLITD	CU
SPLITE	CT
SPLITF	CS
SPLITI	CX
SPLITM	CX
SPLITO	CX
START	CA,CN
TAKECP	CC,CQ
TMINUS	CP
TRUADR	CY
USTOPA	CD,CR
USTOPB	CD,CR
USTOPC	CD,CR
USTOPD	CD,CR
USTOPE	CD,CR
USTOPF	CD,CR
USTOPG	CD
VARMOV	CX
WAY4	CB,CP
WRITE	CL,CX
WTDATA	CL,CX
ZYXWZY	CL

PHASE 3

<u>Label</u>	<u>Chart</u>
ADRTRU	DH,DT
CHKSEQ	DJ,DU
CLOSE	DK,DV
COMP21	DG,DS
COMP31	DG,DS
COMP32	DG,DS
COMP41	DF,DR
COMP42	DF,DR
COMP43	DF,DR
COMP51	DE,DQ
COMP52	DE,DQ
COMP53	DE,DQ
COMP54	DE,DQ
COMP61	DD,DP
COMP62	DD,DP
COMP63	DD,DP
COMP64	DD,DP
COMP65	DD,DP
COMP71	DC
COMP72	DC
COMP73	DC
COMP74	DC
COMP75	DC
COMP76	DC
CONVRT	DJ,DU
CPBPTI	DH,DT
CPBPTO	DJ,DU
CPLINK	DA,DM

EOJSW	DU
EXIT32	DJ, DU
FILL1	DB, DN
FILL2	DB, DN
FILL3	DB, DN
FILL4	DB, DN
FILL5	DB, DN
FILL6	DB, DN
FILL7	DB
FULOUT	DJ, DU
GET1	DB, DN
GET2	DB, DN
GET3	DB, DN
GET4	DB, DN
GET5	DB, DN
GET6	DB, DN
GET7	DB
IF1	DB, DN
IF2	DB, DN
IF3	DB, DN
IF4	DB, DN
IF5	DB, DN
IF6	DB
INSERT	DU
INTLEAVE	DA, DM
INTPH3	DA, DM
IOCSCLAS	DL, DW
IOCSOPEN	DL, DW
ISPLIT	DU
ITCOMP	DA, DM
LABETY	DK, DV
LRMDR	DH, DT
LRMDR1	DH, DT
LWRITE	DJ, DU
LWRITE+4	DJ
MOVVAR	DU
MSPLIT	DU
NXT1R	DG, DS
NXT2R	DG, DS
NXT3R	DG, DS
NXT4R	DF, DR
NXT5R	DE, DQ
NXT6R	DD, DP
NXT7R	DC
NXTOR	DJ
OKLMTS	DH, DT
OPENF	DK, DV
OPTION	DA, DM
OUTAPE	DA, DM
OUTDSK	DA, DM
PH3CON	DA, DM
PH3EJ1	DJ, DU
PH3EOJ	DK, DV
PH3MRG	DA, DM
PRTEOJ	DL, DW
PUT1	DG, DS
PUT2	DG, DS
PUT3	DG, DS
PUT4	DF, DR
PUT5	DE, DQ

PUT6	DD,DP
PUT7	DC
RD1234	DB, DN
RECORD0	DA, DM
RTNEQ	DA, DM
RWLABL	DK, DV
SEQCHK	DJ, DU
SEQERR	DJ, DU
SEQERROR	DL, DW
SPLIT1	DS
SPLIT2	DS
SPLIT3	DS
SPLIT4	DR
SPLIT5	DQ
SPLIT6	DP
SPLITR	DU
START	DA, DM
START1	DA, DM
TAPEO	DJ, DU
TAPLLR	DA, DM
TESTEQ	DA, DM
USRLAB	DL, DW
USTOP1	DB, DN
USTOP2	DB, DN
USTOP3	DB, DN
USTOP4	DB, DN
USTOP5	DB, DN
USTOP6	DB, DN
USTOP7	DB
WTLBLK	DU
X31CLS	DL, DW
X31CLS-12	DL, DW
X31EVT	DL, DW
X31LNK	DL, DW
X31TS1	DL, DW
ZYXWZY-INSERT	DJ

PHASE 4

<u>Label</u>	<u>Chart</u>
ADD1	EJ
ADJOUT	EH
BCERRW	ED
BEGIN	EM
BPASS	ED
CEVCHK	EP
CHKPOINT	EA
CLIFOR	EB
CLOSEI	EN
CLOSEQ	EM
COMP	EM
COMP21	EG
COMP31	EF
COMP32	EF
COMP41	EE

COMP42	EE
COMP43	EE
CONVRT	EH
CPBPTO	EH
CPIADD	EC
DISK	EA
EOF	ED
EOFDK	ED
EOFTAPE	EN
EOJMES	EK
EOVDSK	EH
ERR	ED
ERROR	ED
ERRW	ED
EXIT	EH, EM
FETCH	EA
FEOVOUT	EM
FEOVTP	EN
FILLDL	EB
FIXIP1	EC
FIXLGN	EB
FIXMOV	EH
FLDSKO	EB
FLDSKO+14	EB
FROMDK	ED
FULOUT	EH
GET1	EC
GETAB	EA, EB
GOGO	EB, EM
GOGO1	EN
GOGO2	EN
INIT1	EA, EB
INIT2	EA, EB
INIT3	EA, EB
INIT4	EA
INIT5	EA
INPUT	EM
INSERT	EH
ITCOMP	EB
KEYOK	EB
LALINK	ED
LASTBLK	ED
LMTOKO	EH
LRA	EC
LWRITE	EH
MEND	EN
MODIFY	EN
MULTPL	ED
NIEFI	EN
NIEFO	EM
NOUSER	EB
OKLMTS	EC
OPENA	EM
OPENI	EN
OPENO	EM
OPENX	EM
OUTAPE	EB
OUTDK	EM

OUTFIL	EH
OUTPUT	EM
PANCO	EA, EB
PH3EJ1	EH
PH3EJ2	EJ
PH3EOJ	EJ
PH4EOJ	EK
PROCESS	ED
PROS	EB
PUT1	EG
PUT2	EG
PUT3	EF
PUT4	EE
RESPONSE	EL
RESTOR	ED
RESTOR+4	ED
SEQBACK	EH
SEQERR	EH, EL
SWITCH2	EN
TAPE	EA
TAPE1	EC
TAPE1-2	EC
TAPEG1	EC
TAPEG1-4	EC
TAPEIN	EN
TAPEO	EH
TAPEX	ED
TAPPUT	EH
TAPOFX	EB
TAPOFX+8	EB
TAPOUT	EA, EJ
TAPOV3	EH
TEST	ED
TESTEX	EB
TESTON	EB
TMEOF	ED
TPDKSW	EN
TPOUT	EB
TR1	EC
TR1-4	EC
TSBPASS	EB
TSTFORM	EB
TSTREC	EB
UPDTO	EH
USEREX41	EN
USEREX44	EM
VARMOV	EH
WLR	ED
WLR2	ED
WRITEV	EH
ZYXWZY	EH

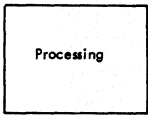
OPTIONAL ROUTINES

<u>Label</u>	<u>Chart</u>
ANICTE	FA
ANIULT	FA
ANIVTA	FA
ANONE	FA
EQINIT	FA
FIXPNT	FB
FIXR2	FB
FIXVTA	FB
FLTBCK	FC
FLTCML	FC
FLTCSG	FC
FLTPNT	FC
FLTSWH	FC
FLTVTA	FC
IGDUM	FF
IGLCOM	FF
IGOUT	FF
PAKCLR	FD
PAKLP2	FD
RELOCA	FA
RLFWB	FA
RLMDGT	FA
RLRUTR	FA
RLSTRP	FA
RLUPDT	FA
ROUT1	FF
SIGCP2	FE
SIGCP4	FD
SIGKR2	FD
SIGPK2	FD
SIGZN2	FE
ZNLST2	FE
ZONPS2	FE

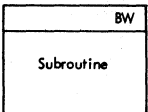
APPENDIX C: EXPLANATION OF FLOWCHART SYMBOLS

DESCRIPTION

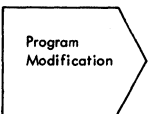
EXAMPLE



A group of program instructions that perform a processing function of the program.



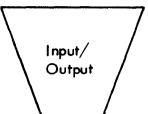
Description or title of a routine that is detailed on another flowchart, which is identified by the letters in the stripe.



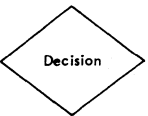
An instruction or group of instructions that changes portions of a routine or initializes a routine for given conditions.



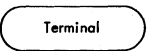
A group of operations not detailed in the flowcharts for this program, such as user's routines.



Any function of an input/output device or program.



Points in the program where branches to alternate processing are made, based upon variable conditions, program switch settings, and test results. Dotted-line decision blocks represent pre-set branches (usually determined at initialization time).



The beginning or end of a program.

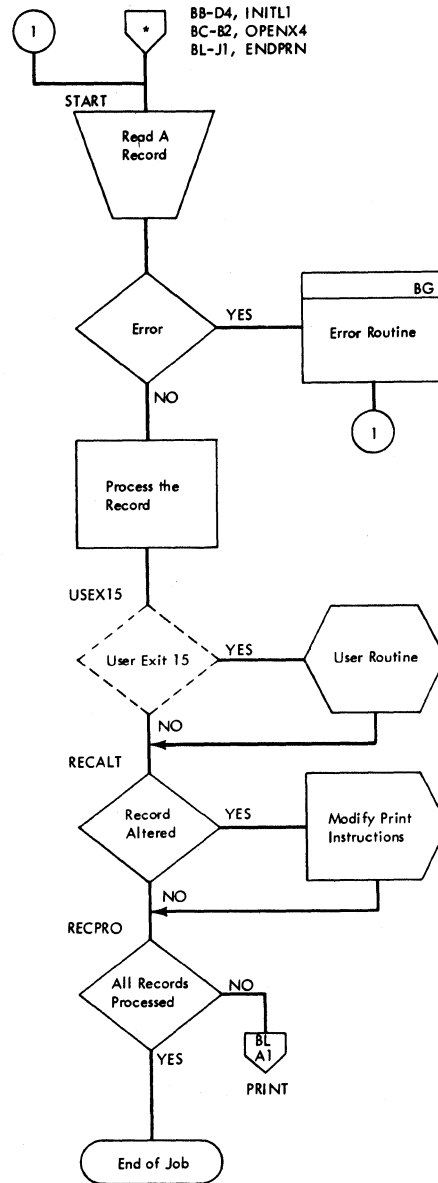


On-page connector. An entry from, or an exit to, another function on the same flowchart. The number in the connector identifies the corresponding entry or exit on the chart.



FILINP

Off-page connector. An entry from, or an exit to, a given point on another flowchart. The characters in the connector identify the chart and block. The corresponding label, if any, is placed outside the connector. For multiple entries and exits, an asterisk appears in the connector and the characters are listed nearby.



Note: Arrowheads are not used on lines that indicate normal flow, which is downward and to the right on page.

APPENDIX D: ERROR MESSAGES

<u>Number</u>	<u>Message</u>	<u>Meaning</u>	<u>Chart</u>
7D01I	COLUMN 1 NOT BLANK. CONTROL CARD NUMBER XX.	Column 1 of a sort/merge Control statement is not blank. <u>XX</u> represents the number of the control statement within the sequence of sort/merge control statements.	AA
7D02I	L3 INVALID FOR ADDRROUT OPTION.	The output record length (L3) must be (1) equal to 10 when ADDRROUT=A, (2) at least 11 when ADDRROUT=D, or (3) no greater than 10 bytes plus the length of all control fields if ADDRROUT=D and exit 32 is not specified.	AG
7D03I	STATEMENT DEFINER INVALID. XXXXXX	The statement definer is invalid or does not appear between columns 2 and 15 in the control card. The definer is identified by <u>XXXXXX</u> .	AA
7D04I	NO END CARD FOUND AFTER READING 25 CONTROL CARDS.	More than 25 control statements have been read without encountering an END statement. The maximum number of control statements permitted is 25.	AA
7D05I	CONTINUATION CONTROL CARD XX DOES NOT START IN COLUMN 16.	A continuation card must begin in column 16. <u>XX</u> denotes the number of the invalid control statement.	AA
7D06I	BLANK CONTROL CARD DETECTED.	A blank control card has been intermixed with the sort/merge control statements.	AA
7D07I	MANDATORY XXXXXX CARD OMITTED.	A mandatory control statement has been omitted. The statement definer of the missing card is identified by <u>XXXXXX</u> .	AB
7D08I	TYPE RUN NOT KNOWN - SORT OR MERGE NOT SPECIFIED.	Neither a SORT nor a MERGE control statement was included.	AB
7D09I	NO BLANK AFTER STATEMENT DEFINER - XXXXXXX	A blank does not separate the statement definer from the first operand definer. The first six X's identify the statement definer; the last X identifies the invalid punched character.	AC, AD

<u>Number</u>	<u>Message</u>	<u>Meaning</u>	<u>Chart</u>
7D10I	FIELD DEFINER INVALID - XXXXXXXX.	The operand definer identified by <u>XXXXXXX</u> is not recognized as a valid operand definer.	AC, AD
7D11I	VALUES INVALID - XXXXXX.	The value(s) following an operand definer is invalid. <u>XXXXXX</u> identifies the invalid value(s).	AC, AD
7D12I	INVALID FORMAT CODE GIVEN - XX.	The format code for the input data is punched incorrectly.	AC
7D13I	SORT AND MERGE CONTROL CARDS SPECIFIED IN SAME RUN.	Both a SORT and a MERGE control statement were included. Only one is acceptable.	AC
7D14I	NO SEQUENCE VALUE GIVEN FOR CF XX.	No sequence (ascending or descending) has been specified in the SORT or MERGE control statement for one or more control data fields.	AC
7D15I	MORE THAN 12 CONTROL FIELDS SPECIFIED.	The maximum number of control fields to be used in sorting or merging is 12.	AC
7D16I	FORMAT ENTRY NOT SPECIFIED.	The FORMAT operand definer is not specified in either a SORT or MERGE control statement.	AG
7D17I	NO MAJOR CONTROL FIELD WAS GIVEN.	Control field 1 specifications are not recognizable to the program because the FIELDS operand definer was not included in a SORT or MERGE control statement.	AG
7D19I	FIXED BLOCKING SPECIFIED FOR VARIABLE LENGTH RECORDS.	Variable-length records on input must be specified as being in variable-length blocks.	AG
7D20I	CONTROL FIELD XX EXTENDS BEYOND END OF RECORD.	A control data field identified by <u>XX</u> has been specified beyond the last valid byte of the logical record.	AJ
7D21I	TOTAL LENGTH OF CONTROL FIELDS EXCEEDS 256.	The total length of all control data fields is limited to a maximum of 256 bytes.	AG
7D22I	CONTROL FIELD XX GREATER THAN MAXIMUM ALLOWED.	The control data field identified by <u>XX</u> exceeds (1) 16 bytes for a decimal field, or (2) 4 or 8 bytes for a normalized floating-point number.	AJ

<u>Number</u>	<u>Message</u>	<u>Meaning</u>	<u>Chart</u>
7D23I	L4 MUST BE LESS THAN (L1) (L5)	When sorting variable-length records, the minimum input record length must be less than the maximum or average input record length.	AG
7D24I	STORAGE SPECIFIED GREATER THAN ACTUAL MACHINE SIZE.	The value specified in the STORAGE entry is greater than the machine size specified at IPL time.	AG
7D25I	(L3) (L1) MORE THAN XXXX BYTES.	The input or output record length exceeds the maximum record length acceptable to the sort/merge program.	AJ
7D26I	KEYLEN ENTRY INVALID.	The KEYLEN operand definer can only be specified for fixed-length unblocked records (disk input only).	AJ
7D27I	STORAGE SPECIFIED MORE THAN 524,288 BYTES.	The maximum main storage capacity supported by the sort/merge program is 512K.	AG
7D28I	RECORD TYPE NOT SPECIFIED.	The TYPE operand definer used to indicate fixed- or variable-length records has not been specified.	AC
7D29I	FILES ENTRY NOT SPECIFIED FOR MERGE.	The number of files to be merged has not been specified. The FILES entry is mandatory for a merge-only operation.	AD
7D30I	SIZE ENTRY OMITTED IN SORT STATEMENT.	The SIZE operand definer is mandatory for a sort operation. It is used to reflect the exact or approximate number of records to be sorted.	AJ
7D32I	USER PROGRAM ORIGIN GREATER THAN STORAGE SIZE.	The main storage load point or origin address for a user program has been specified beyond the boundaries of the storage size. All user programs must be loaded below the storage size indicated at IPL time or in the STORAGE entry.	AG
7D33I	L5 IS GREATER THAN L1.	L5 has been specified greater than L1 for variable-length records. L5 must be specified as the average logical record length, or as a value between the average and the maximum (L1).	AG

<u>Number</u>	<u>Message</u>	<u>Meaning</u>	<u>Chart</u>
7D34I	(E32) (E43) NOT SPECIFIED WHEN L3 (MORE) (LESS) THAN L1.	If L3>L1, exit 32 or exit 43 must be included to accomplish record lengthening in phase 3 or phase 4. If L3<L1 and variable-length records have been specified, exit 32 or exit 43 must be used to update the record length field of each truncated record.	AG, AJ
7D35I	EXIT (31) (44) NOT SPECIFIED FOR NONSTANDARD LABELS.	When nonstandard output tape labels have been specified, it is the user's responsibility to create and write them. This function is performed through exit 31 or exit 44.	AJ
7D36I	USER GIVEN FILE SIZE EXCEEDS MAXIMUM.	The specified sort work area allocated in the FILEW extent cards is not large enough to process the file size specified in the SIZE entry of the SORT control statement.	AP
7D37I	INPUT BLOCKSIZE NOT A MULTIPLE OF L1.	The number of bytes in an input block for fixed-length records must be a multiple of the number of bytes in each input record.	AJ
7D38I	OUTPUT BLOCKSIZE NOT A MULTIPLE OF L3.	The number of bytes in an output block for fixed-length records must be a multiple of the number of bytes in each output record.	AJ
7D39I	A CF STARTS PRIOR TO BYTE 5 IN VARIABLE-LENGTH RECORDS.	The first four bytes of a variable-length record are the record-length field and must not be used as a control data field.	AJ
7D40I	CONTROL FIELDS OVERLAP FOR OTHER THAN BI FORMAT.	Overlapping control data fields are valid only with the unsigned binary data format.	AJ
7D41I	RECORD LENGTH NOT SPECIFIED.	The operand definer LENGTH or its value (L1) has not been specified.	AG
7D42I	BLOCKSIZE GREATER THAN XXXX.	The input or output block length specified is greater than the maximum acceptable to the program.	AJ
7D43I	NOTPMK ENTRY SPECIFIED WITH STANDARD OUTPUT LABELS.	The NOTPMK entry is valid only for unlabeled tape output files or for tape output files with nonstandard labels.	AJ

<u>Number</u>	<u>Message</u>	<u>Meaning</u>	<u>Chart</u>
7D44I	PHASE (1) (3) (4) MODIFICATION PROGRAM TOO LARGE.	The size of the user program (determined by the <u>Address</u> value in the MODS statement) is such that it forces the sort block size below the required minimum.	AH, AK, AM, AN
7D45I	NO MEDIUM SPECIFIED FOR (INPUT) (OUTPUT).	The type of input or output medium (tape or disk) has been omitted from the INPFIL or OUTFIL control statement.	AG
7D47I	(TAPE) (DISK) OPTIONS SPECIFIED FOR (DISK) (TAPE) (INPUT) (OUTPUT).	Tape options such as OPEN and CLOSE can only be specified for tape files. Disk option such as KEYLEN and VERIFY pertain only to disk files.	AJ
7D49I	NO BLOCKSIZE GIVEN FOR (INPUT) (OUTPUT).	The operand definer BLKSIZE has either been incorrectly specified or omitted.	AG
7D50I	INSUFFICIENT TRACKS GIVEN FOR MERGE.	A minimum of 2 adjacent disk tracks must be allocated for the work area in a merge-only operation.	AF
7D51I	ADDRROUT OPTION SPECIFIED FOR MERGE.	The ADDRROUT option cannot be specified for a merge-only operation.	AJ
7D53D	INVALID RESTART.	A restart sort run has been specified, but the original sort was interrupted prior to the end of phase 1.	AS
7D55A	INVALID RESTART. CHECK DISK PACK PLACEMENT.	1. The disk pack(s) which contains the sort work area was not placed on a drive assigned to the identical symbolic unit used in the initial run, or 2. The sort data has been destroyed since the original job.	AS
7D64I	DUPLICATE STATEMENT DETECTED - XXXXXXXX.	Two control statements contain identical statement definers.	AA
7D67I	INVALID LABELS SPECIFIED FOR A DISK FILE.	Disk input or disk output has been specified, and the labels associated with the file(s) have not been specified as standard. All disk files must contain standard labels.	AJ

<u>Number</u>	<u>Message</u>	<u>Meaning</u>	<u>Chart</u>
7D68I	(INPUT) (OUTPUT) BLOCKSIZE INVALID FOR VARIABLE- LENGTH RECORDS.	The input or output block size specified is less than the maximum input record length plus four bytes. The input or output block size must be equal to or greater than L1 + 4.	AJ
7D69I	SORT BLOCKSIZE MUST BE AT LEAST 300 BYTES.	The size (total number of bytes) of a user program in phase 1 or phase 3 has forced the assignment phase to compute a sort block size that is less than 300 bytes.	AJ
7D70I	INPUT OR OUTPUT BLOCKSIZE IS INVALID.	The input or output block size specified for a merge-only run exceeds the maximum size allowed.	AH
7D79I	BLOCKSIZE FOR TAPE INPUT OR OUTPUT IS LESS THAN 12.	The minimum input and output block size for tape operations is limited to 12 bytes.	AJ
7D81I	EXIT 13 SPECIFIED FOR DISK INPUT.	Exit 13 can only be specified in a sort operation when tape input has been specified.	AJ
7D82I	ADDRROUT OPTION SPECIFIED WITH TAPE INPUT.	The ADDRROUT option can only be specified for a sort run when disk input has been specified.	AJ
7D83A	INVALID RESPONSE.	An invalid response to message 7D53D, 7D55A, or 7D90A has been received from the operator.	AA, AS
7D84I	TAPE DEVICE ADDRESSES MUST BE ASSIGNED TO (SYSXXX) (SYSnnn).	For a sort operation, all tape input files must reside on SYS002-SYS010, depending on the number of files to be sorted. For a merge-only operation, tape FILEA must be on SYS002, tape FILEB must be on SYS003, etc. For tape output, SYS001 must be the output unit. The listed symbolic units do not have tape device addresses assigned to them.	AJ
7D85I	ALL TAPE FILES MUST HAVE UNIQUE DEVICE ADDRESSES.	The message can occur only during a merge-only run. At least two tape files (input and output) reside on symbolic units with an identical device address.	AJ

<u>Number</u>	<u>Message</u>	<u>Meaning</u>	<u>Chart</u>
		For tape input and/or output, all tape files must reside on different tape drives. For instance, for a 2-way tape merge, FILEA must reside on SYS002; FILEB must reside on SYS003; SYS002 and SYS003 must have been assigned to different tape device addresses. If tape output is specified, SYS001 must be a tape device other than SYS002 and SYS003.	
7D90A	OPERATOR - ATTEMPT TO CORRECT ABOVE LISTED ERRORS.	This message occurs at the end of the assignment phase when errors have been detected and both SYSRDR and SYSIPT are card readers. It applies to all assignment phase diagnostic messages except 7D53D, 7D55A, 7D83A, and 7D92I. This facility is provided to enable the sort/merge program to be executed when it is only a job step within a specific job application. If the errors can be corrected immediately, the operator should do so.	AA
7D92I	END OF ASSIGNMENT PHASE - ERRORS DETECTED, CORRECT AND RERUN	Errors have been detected and listed by the assignment phase. SYSRDR and/or SYSIPT are not card readers or SYSLOG is not a printer-keyboard.	AA
7DA1I	WLR - FILEX	Phase 1 has detected a wrong-length record (block) during a read operation. X indicates the file from which the wrong-length record was read. This message can occur when the records in the input file are not the same length as those specified in the L1 value of the RECORD statement or the input BLKSIZE entry was specified incorrectly.	BD, BE, BK
7DA2I	PHASE 1 UNREADABLE BLOCKS BYPASSED XXXX.	This message is printed at the end of phase 1 when tape input has been specified, and either the BYPASS option or exit 13 has been specified. The message indicates the number of input blocks bypassed by the sort.	BE

<u>Number</u>	<u>Message</u>	<u>Meaning</u>	<u>Chart</u>
7DA3I	WORK AREA TOO SMALL FOR ACTUAL FILE	The work area specified in the FILEW extent card(s) is not large enough to process the number of records contained in the input file(s).	BH
7DC2A	INVALID RESPONSE.	An invalid response has been received in reply to message 7DC2D.	DL
7DC2D	SEQ. ERROR.	This message should never occur. However, when it does, it is to be interpreted as a program error. A sequence error has been detected during the merging process in phase 3.	DJ
7DD1I	WLR FILE X	Phase 4 has read a wrong- length record. <u>X</u> represents the file from which the wrong-length record was read. (See 7DA1I for further explanation.)	ED
7DD2A	INVALID RESPONSE	An invalid response has been received in reply to message 7DD2D.	EL
7DD2D	SEQ. ERROR FILE X	A sequence error has been detected in phase 4. <u>X</u> identifies the file with the sequence error. This message can occur either because the file was not pre-sequenced or the control data information was incorrectly specified in the MERGE control statement.	EH
7DD4I	PHASE 4 UNREADABLE BLOCKS BYPASSED 00000.	This message indicates the number of input blocks bypassed during phase 4 when either the BYPASS option or exit 45 has been specified.	ED

- Bucket - A location in main storage used for temporary storage of constants or values that are to be used again later in the program.
- Doublet - A sequence of only two records.
- Formatting - The division of a disk work area into portions which are assigned for specific purposes.
- Interleave factor - A calculated value, based on the order of merge, used to determine the gap length in the disk work area when interleaving output blocks.
- Interleaving - The output of merged blocks to a disk work area according to a pattern that minimizes seek time during succeeding passes.
- No-op - No-operation instructions. Used as a switch to branch to another group of instructions, or to another routine, when switch is turned on by other instructions in the program. Usually coded as 'NOP' or 'BC 0, operand.' The no-op is made a branch by moving a hexadecimal 'F0' into the first operand of the instruction.
- Order of merge - The number of input sequences that are merged into one output sequence.
- Pass - The processing of an input file from its first to its last logical record.
- Sequences - A group of records in a specified sequential order. Also referred to as strings.
- Set - The number of sequences equal to the order of merge.
- Strings - See sequences.
- Winner - The record that is selected to be moved to the output area as the result of a comparison or a series of comparisons. The "winning" record is the one that is determined to be the next one to fit in the sequence currently in the output area.

INDEX

ABORT	29	CKBMAX	45
ADD1	187	CKBPT	52
ADDROUT Run, Initialization	81	CKBYPTKR	59
ADJOUT	186	CKDEFS	34
Alter Record Length (Phase 4)	174	CKFLD1	34
ANIULT	195	CKFLD4	34
ANIVTA	195	CKKEY	49
Assignment Phase (Phase 0)	12, 23	CKL3MAX	43
		CKL3MX	46
B1	44	CKLABELS	48
B118	63	CKLMAX	43
B119	63	CKMERGE	49
B126	63	CKPARA	27
B135	63	CKPRINT	65
B136	63	CKPTOK (FLR)	100
B170	63	CKPTOK (VLR)	117
B2	44	CKRECBK	55
BCERRW	179	CKRECBKR	61
BEGIN (Phase 0)	62	CKRPT	61
BEGIN (Phase 4)	189	CKRPT1	61
BEGIN1	25	CKRWD	48
BLDADR	83	CKSIZ	48
BLK2	40	CKTYPEIN	49
BLKHIP	48	CKVOL	49
BLKOK	48	CLEAR (DSORT005)	36
BLMAX4	42	CLEAR (DSORT008)	51
BPASS	179	CLIFOR	175
BRCH	65	CLORBV	42
BSTR6 (FLR)	110	CLOSE (FLR)	146
BSTR6 (VLR)	128	CLOSE (VLR)	164
BYPAS1+4 (FLR)	103	CLOSEI	192
BYPAS1+4 (VLR)	120	CLOSEO	190
BYPVAR	54	CLUB	62
BYWCKD (FLR)	98	CMPCHN	76
BYWCKD (VLR)	113	CMPRES	81
		COL16	26
C	48	COMBPT	52
CALADR (FLR)	110	COMBPTV	56
CALADR (VLR)	128	COMP	32
CALNOP	57	COMP (Phase 0)	32
CALPH1B2	51	COMP (Phase 4)	189
CALRPT	55	COMP21	
CALSW2V	57	Phase 3 (FLR)	141
CBYPTK	53	Phase 3 (VLR)	156
CBYPTKV	56	Phase 4	182
CCDESC	65	COMP32	
CCERR	29	Phase 3 (FLR)	141
CCS0	30	Phase 3 (VLR)	156
CF256	40	Phase 4	181
CFB3MAX	45	COMP43	
CFL3MAX	43	Phase 3 (FLR)	139
Checking Features	20	Phase 3 (VLR)	156
Checkpoint, Interrupt, and Restart	20	Phase 4	181
Checkpoint Routine (Phase 4)	192	COMP54 (FLR)	139
CHKPOINT	171	COMP54 (VLR)	154
CHKPT	72	COMP65 (FLR)	139
CHKSEQ (FLR)	144	COMP65 (VLR)	154
CHKSEQ (VLR)	159	COMP76	138
CHNCMP	82	COMPBA (FLR)	108
CHNMVE	83	COMPBA (VLR)	125
CKAMAX	57	COMPGB (FLR)	107
CKBI	44	COMPGB (VLR)	124

COMPDC (FLR)	106	DSORT	25
COMPDC (VLR)	123	DSORT002	30
COMPED (FLR)	105	DSORT003	31
COMPED (VLR)	121	DSORT004	33
COMPFE (FLR)	104	DSORT005	36
COMPFE (VLR)	120	DSORT006	42
COMPG	53	DSORT007	45
COMPGF	103	DSORT008	51
COMPGR	60	DSORT009	61
COMPGV	56	DSORT010	65
COMPIT (FLR)	97	E181	63
COMPIT (VLR)	112	E200	64
Compute Constants		ENDGF	54
ADDROUT Run	58	ENDGFR	60
FLR	50	ENDINP	85
VLR	55	ENDMAC	85
Compute Maximum I/O Record and Block		End of Job Messages (Phase 4)	187
Sizes	42	End of Job Routine (Phase 4)	187
CONOUT	66	End-of-Phase Routine (Phase 1)	80
Constants (Appendix A)		ENDPAS (FLR)	99
Phase 0	301	ENDPAS (VLR)	116
Phase 1	317	ENDSCN	34
Phase 2	323	EOF	179
Phase 3	332	EOFADDR	85
Phase 4	341	EOFDK	178
Continuation Cards	28	EOFINIT	71
Control Card Scan Routines	31, 33	EOFTAPE	192
Control-Data Fields	17	EOJ	66
Convert/Reconvert Routines		EOJ3RT	67
Fixed-Point	196	EOJC1	67
Floating-Point	196	EOJCALC	66
Packed-Decimal	197	EOJMES	187
Zoned-Decimal	197	EOJSW	160
CONVRT	183	EOJZ	66
Phase 3 (FLR)	144	EOVDSK	186
Phase 3 (VLR)	160	EOVEOF	85
Phase 4	183	EQINIT	195
CORVAL	39	Equal Routine	198
CPBPTI (FLR)	143	ERR	179
CPBPTI (VLR)	157	ERR1	26
CPBPTO (FLR)	145	ERR3	28
CPBPTO (VLR)	163	ERR3A	28
CPIADD	177	ERR4	26
CPIOAS (FLR)	97	ERR5	26
CPIOAS (VLR)	113	ERR6	26
CPLINK (FLR)	135	ERR64	27
CPLINK (VLR)	150	ERROR	179
CPOADR-12	118	Error Messages (Appendix D)	365
CPOADR-8	100	Error Routine (Phase 4)	178
CVB3MAX	45	ERRW	180
CVL3MAX	43	EXECPP (FLR)	100
		EXECPP (VLR)	117
Data Conversion	19	EXIT	
DCLOSE	85	Phase 0	38
Definitions of Abbreviations	349	Phase 1	77
DETLLOWER	44	Phase 4 (DSORT401)	189
DETSMALL	43	Phase 4 (DSORT402)	183
DIBLRD	74	Exit 11 Linkage Routine	84
DINGET	83	Exit 11 Linkage Routine, Initialization	71
DINGUD	83	EXIT32 (FLR)	144
DINLES	83	EXIT32 (VLR)	160
DINMVA (ADDROUT Run)	83	EXITP1	81
DINMVA (Sort Run)	74	External Sort or Merge Phase	
DISK	171	(Phase 2)	12, 86
Disk Input Routine (Phase 1)	74	FEOVBR	85
DIVAGN (FLR)	97	FEOVOUT	190
DIVAGN (VLR)	113		
DSKINIT	71		

FEOVRTN	85	IBVMC3	79
FEOVTP	191	IBVMH4	79
FETC10	64	IBVNE4	80
FETCH		IBVPB2	79
Phase 0	29	IBVVLR	80
Phase 1 (DSORT102)	72	IGLCOM	198
Phase 1 (DSORT105)	81	ILEAVE-10 (FLR)	101
Phase 4	171	ILEAVE-10 (VLR)	118
FETCH1	67	INEXIT (ADDROUT Run)	84
FETCH2	67	INEXIT (Sort Run)	75
FETCH4	67	INF	66
FETCH7	45	INF1	33
FETCH9	55, 58	INFINI (ADDROUT Run)	84
Fetch Next Phase	64	INFINI (Sort Run)	74
File B Compare Loop (Phase 4)	182	INFSCN	33
File C Compare Loop (Phase 4)	181	INIT1	
File D Compare Loop (Phase 4)	180	Phase 0	37
FILLA (FLR)	102	Phase 4 (DSORT401)	169
FILLA (VLR)	119	Phase 4 (DSORT402)	172
FILLDL	176	INIT2	
FINAL	64	Phase 0	38
Final Merge Phase (Phase 3)	12, 131	Phase 4 (DSORT401)	170
FIRSTED	39	Phase 4 (DSORT402)	172
FIXED	62	INIT3 (DSORT401)	170
Fixed-point Convert/Reconvert Routine	196	INIT3 (DSORT402)	175
FIXIP1	177	INIT32	72
FIXLGN	174	INIT4	170
FIXMOV	185	INIT5	170
FIXPNT	196	INITA1	38
FIXR2	196	INITEX11	71
FIXVTA	196	Initialization Routines	
FLDSKO	175	Phase 1, ADDROUT Run	81
FLDSKO+14	175	Phase 1, Sort Run	72
FLMVBC	73	Phase 1, Multi-Volume	71
FLMVGO	73	Phase 2, FLR	96
Floating-point Convert/Reconvert		Phase 2, VLR	110
Routine	196	Phase 3, FLR	133
Flowchart Symbols	364	Phase 3, VLR	148
FLTBACK	196	Phase 4, DSORT401	169
FLTPNT	196	Phase 4, DSORT402	171
FLTVTA	196	INITOM (FLR)	101
FORMAT	72	INITOM (VLR)	118
Format Routine	72	INITST	27
FORMGO	72	INMOVE	74
FROMDK	179	INPTIZ	73
FULOUT		INPUT	190
Phase 3 (FLR)	144	Input Routines	
Phase 3 (VLR)	161	Phase 1, ADDROUT Run	82
Phase 4	185	Phase 1, Disk or Tape	74
GET1	177	Phase 1, Tape Only	75
GETA (FLR)	102	Phase 2, FLR	101
GETA (VLR)	119	Phase 2, VLR	119
GETAB	169	Phase 3, FLR	136
GETL1	45	Phase 3, VLR	151
GETXT	85	Phase 4	176
GOGO	173	INSERT (FLR)	162
GOGO1	190	INSERT (VLR)	185
GOGO2	191	INTAPE	71
IBV021	72	Interleaved Disk Address Routine	
IBV022	73	Phase 2 (FLR)	110
IBV023	82	Phase 2 (VLR)	128
IBV024	81	Phase 3 (FLR)	141
IBVAA1 (ADDROUT Run)	82	Phase 3 (VLR)	157
IBVAA1 (Sort Run)	73	Interleaved Output Technique	13, 90
IBVIF4	76	Internal-Sort Phase (Phase 1)	12, 68
IBVJC2	77	Internal Sort Routine (ADDROUT Run)	84
		Internal Sort Routine (Sort Run)	75
		INTPH2 (FLR)	96

INTPH2 (VLR)	112	LWRITE	186
INTPH3 (FLR)	133	Phase 3, FLR	145
INTPH3 (VLR)	149	Phase 3, VLR	162
INTLEAVE (FLR)	134	Phase 4	186
INTLEAVE (VLR)	149	Machine Requirements	22
Introduction	9	Mainline Compare Routine,	
IOCSCLOS (FLR)	148	Phase 3 (FLR)	138
IOCSCLOS (VLR)	165	Mainline Compare Routine,	
IOCSOPEN (FLR)	147	Phase 3 (VLR)	153
IOCSOPEN (VLR)	164	MEND	192
IRMDR (FLR)	103	Merge-Merge Routine (FLR)	98
IRMDR (VLR)	120	Merge-Merge Routine (VLR)	114
IRMDR1 (FLR)	103	Merge-Only Phase (Phase 4)	12, 167
IRMDR1 (VLR)	120	METH1I	37
ISMERG	47	MGINIT	36
ISPLIT	161	MINBLKI	47
ITCOMP		MMPP (FLR)	109
Phase 3 (FLR)	133	MMPP (VLR)	127
Phase 3 (VLR)	149	MMPP1 (FLR)	110
Phase 4	175	MMPP1 (VLR)	129
Job Control Statements	13	MMPP2 (FLR)	98
KEYOK	172	MMPP2 (VLR)	114
L1	42	MMPPS (FLR)	99
L1MULT	48	MMPPS (VLR)	115
L3L1	46	MNLDSK	84
Label Linkage Routine (FLR)	147	MNLLINK	85
Label Linkage Routine (VLR)	164	MNLTAP	85
Label Processing	20	MOD1	34
Label References (Appendix B)		MODIFY	191
Optional Routines	363	MODSCN	34
Phase 0	350	MOPSI	79
Phase 1	354	MOVECC	30
Phase 2	356	MOV MVC	109
Phase 3	358	MOVVAR	161
Phase 4	360	MRGCKP	38
LABETY (FLR)	146	MRGOPN	37
LABETY (VLR)	164	MRGSCN	32
LABEXCK	49	MSG91	67
LALINK	180	MSG91A	67
LASTBLK	180	MSPLIT	161
LBLCHK	84	Multi-volume, Exit 11 Linkage	84
LBLINIT	71	Multi-volume Initialization	71
LENHI	46	MULTPL	180
LENLO	46	NEWITL (FLR)	101
Level 1 (Internal Sort)	75	NEWITL (VLR)	118
Level 2 (Internal Sort)	75	NEXTPASS (FLR)	101
LM1 (FLR)	102	NEXTPASS (VLR)	118
LM1 (VLR)	119	NIEFI	191
LM1234 (FLR)	99	NIEFO	189
LM1234 (VLR)	115	NOCALCS	65
LMO (FLR)	109	NOCHG (FLR)	99
LMO (VLR)	128	NOCHG (VLR)	117
LMOSW (FLR)	99	NOEOB	84
LMOSW (VLR)	116	NORAF	45
LMTSOK (FLR)	110	NOTLAS (FLR)	101
LMTSOK (VLR)	128	NOTLAS (VLR)	118
LOAD1	29	NOUSER	176
LOCARE	82	NOVAR	42
LOOP	46	NOWCKD (FLR)	98
LRA	177	NOWCKD (VLR)	114
LRMDR (FLR)	143	NXTAR (FLR)	108
LRMDR (VLR)	157	NXTAR (VLR)	126
LRMDR1 (FLR)	143	NXTBR (FLR)	108
LRMDR1 (VLR)	157	NXTBR (VLR)	125
		NXTCR (FLR)	107

NXTCR (VLR)	124	OV2INW	81
NXTDR (FLR)	106	Overlay Linkage (Phase 4)	193
NXTDR (VLR)	123	OVLPCF	48
NXTER (FLR)	105		
NXTER (VLR)	122	PLINBG (ADDROUT Run)	83
NXTFILE	85	PLINBG (Sort Run)	74
NXTFR (FLR)	105	PLLEV1	76
NXTFR (VLR)	121	PLSIZE	40
NXTGR	104	P2SIZE	40
NXTOR	144	P3SIZE	40
NXTREC	83	P4NAM	40
		P4SIZ	40
OKLMTS	177	Packed Decimal (SIGPAK) Convert/Reconvert	
OLEAVE (FLR)	101	Routine	197
OLEAVE (VLR)	118	Padding Routine, Phase 1	79
ONEXIT (ADDROUT Run)	84	PADRTN	75
ONEXIT (Sort Run)	75	PAKLP2	197
OPEN	36	PANCO (DSORT401)	169
OPENA (Phase 1)	84	PANCO (DSORT402)	172
OPENA (Phase 4)	189	Pass-Pass Routine, Fixed-length	
Open/Close Routine (Phase 4)	188	Records	100
OPENF (FLR)	146	Pass-Pass Routine (FLR)	100
OPENF (VLR)	164	Pass-Pass Routine (VLR)	117
OPENI	190	PH3CON (FLR)	133
OPENT	85	PH3CON (VLR)	149
Open Work Area and Pre-edit	35, 38	PH3EJ1	
OPENX	189	Phase 3 (FLR)	146
OPERROR	29	Phase 3 (VLR)	163
OPFLE	66	Phase 4	186
OPTION (FLR)	133	PH3EJ2	187
OPTION (VLR)	149	PH3EOJ	187
Optional Routines	194	Phase 3 (FLR)	146
OPTSC1	28	Phase 3 (VLR)	163
OPTSC2	29	Phase 4	187
OPTSCN	34	PH3MRG (FLR)	135
OPTSCS	28	PH3MRG (VLR)	151
ORMDR (FLR)	109	PH4EOJ	187
ORMDR (VLR)	128	Phase 0	12, 23
ORMDR1 (FLR)	109	Phase 1	12, 68
ORMDR1 (VLR)	128	Phase 2	12, 86
OSPLIT	129	Phase 3	12, 131
OTFSCN	34	Phase 4	12, 167
OUP TOK (FLR)	97	Post-Edit Routine	45
OUP TOK (VLR)	113	Print Control Card and Fetch Next Phase	64
OUTAPE		PRNTCARD	65
Phase 3 (FLR)	135	Program Characteristics	13
Phase 3 (VLR)	151	Program Generation	12
Phase 4	175	Program Organization	12
OUTDK	190	PROS	174
OUTDSK (FLR)	134	PRTCALCS	65
OUTDSK (VLR)	150	PRTEOJ (FLR)	148
OUTF2	66	PRTEOJ (VLR)	165
OUTFIL	183	PUT1	
OUTFUL (FLR)	109	Phase 3 (FLR)	141
OUTFUL (VLR)	126	Phase 3 (VLR)	156
OUTIZ (ADDROUT Run)	82	Phase 4	182
OUTIZ (Sort Run)	73	PUT2	182
OUTPT	55	PUT3	182
OUTPUT	189	PUT4	181
Output Routines		PUTA (FLR)	108
Phase 1, ADDROUT Run	84	PUTA (VLR)	125
Phase 1, Sort Run	77	PUTB (FLR)	108
Phase 2, FLR	108	PUTB (VLR)	125
Phase 2, VLR	126	PUTC (FLR)	107
Phase 3, FLR	143	PUTC (VLR)	124
Phase 3, VLR	159	PUTCNSWR	60
Phase 4	182	PUTCO2	55

PUTCO2R	61	SEQBACK	183
PUTCON	54	SEQCHK (FLR)	144
PUTCOV	57	SEQCHK (VLR)	159
PUTD (FLR)	106	SEQERR	
PUTD (VLR)	123	Phase 3 (FLR)	144
PUTE (FLR)	105	Phase 3 (VLR)	159
PUTE (VLR)	122	Phase 4 (DSORT401)	188
PUTF (FLR)	104	Phase 4 (DSORT402)	183
PUTF (VLR)	121	SEQERROR (FLR)	148
PUTG	104	SEQERROR (VLR)	166
		Sequence B Compare Loop (FLR)	107
RAFBI	44	Sequence B Compare Loop (VLR)	125
RAFRTN	58	Sequence C Compare Loop (FLR)	107
RC1	32	Sequence C Compare Loop (VLR)	123
RCDSN	32	Sequence Checking (Phase 4)	183
RCHKPT (FLR)	96	Sequence D Compare Loop (FLR)	106
RCHKPT (VLR)	112	Sequence D Compare Loop (VLR)	122
RDABCD (FLR)	102	Sequence E Compare Loop (FLR)	105
RDABCD (VLR)	119	Sequence E Compare Loop (VLR)	121
RDCPOK (FLR)	96	Sequence Error Routine (Phase 4)	188
RDCPOK (VLR)	112	Sequence F Compare Loop (FLR)	104
RE1	65	Sequence F Compare Loop (VLR)	120
Read and Compress Control Cards	25	Sequence G Compare Loop (FLR)	103
READX	37	SHTPTR (FLR)	101
READX1	37	SHTPTR (VLR)	118
READXT	37	SIL-4	37
Record Format	16	SIGCP2	198
RECORD0 (FLR)	134	SIGCP4	197
RECORD0 (VLR)	149	SIGKR2	197
REDOUT	77	SIGPK2	197
REDUCE	52	SIGZN2	198
REDUCEI (FLR)	99	Sort Compressed Control Cards	30
REDUCEI (VLR)	115	Sort/Merge Control Statements	14
REDUCEO (FLR)	99	SORTXT	37
REDUCEO (VLR)	115	SPADE	62
REDUCER	59	SPLIT6	156
REDUCEV	57	SPLITA	126
REINIT	55	SPLITB	125
RELOCA	194	SPLITC	125
Relocator Routine	194	SPLITD	123
REPL	63	SPLITE	122
REPB	63	SPLITF	121
RESPONSE	188	SPLITI	127
RESTOR (Phase 0)	64	SPLITM	127
RESTOR (Phase 4)	180	SPLITO	127
RETRY (DSORT008)	54	SPLITR	162
RETRY (DSORT009)	62	SRT1	31
RLFWB	194	SRTSCN	31
RLMDGT	195	START	97
RLRUTR	194	Phase 2 (FLR)	97
RLSTRP	195	Phase 2 (VLR)	113
RLUPDT	195	Phase 3 (FLR)	135
ROUT1	198	Phase 3 (VLR)	150
RTNEQ (FLR)	135	START1 (FLR)	135
RTNEQ (VLR)	150	START1 (VLR)	151
RUNCODSW	26	Statement Definers	27
RWLABL (FLR)	146	STBO	62
RWLABL (VLR)	164	STDFV	40
		STEP	51
S188	64	STEP1	51
SAVCKP	38	STEP1B	51
SCAN	32	STEP1BR	58
Scan INPFIL, OUTFIL, MODS, OPTION and		STEP1R	58
END Control Cards	33	STEP1V	55
Scan Sort, Merge, and Record Control		STEP3	52
Cards	31	STEP3A	52
SEEKTM	64	STEP4	52
Select Order of Merge	61	STEP4R	59

STORM1	76	USEREX41	191
STORM2	76	USEREX44	189
STVSOM	63	User Modification and Exits	19, 20
SVORIG	58	USESTR	40
SW	26	USRLAB (FLR)	147
SW1 (DSORT)	27	USRLAB (VLR)	165
SW1 (DSORT008)	53	USTOP1 (FLR)	136
SW1R	61	USTOP1 (VLR)	152
SW2V	58	USTOP2 (FLR)	137
SWITCH2	191	USTOP2 (VLR)	152
Symbolic Unit Assignments	21	USTOP3 (FLR)	137
Symbols for Flowcharts	364	USTOP3 (VLR)	153
System Requirements	22	USTOP4 (FLR)	137
		USTOP4 (VLR)	153
TABLEF	64	USTOP5 (FLR)	137
TAGADR	83	USTOP5 (VLR)	153
TAGCHK	82	USTOP6 (FLR)	138
TAGINT	82	USTOP6 (VLR)	153
TAGMNL	83	USTOP7	138
TAGOUT	62	USTOPA (FLR)	102
TAKECP (FLR)	100	USTOPA (VLR)	119
TAKECP (VLR)	117		
TAPE	171	VALIDATE	47
TAPE1	176	VARBI	44
TAPE1-2	176	VARBLK	83
TAPEG1	177	VARMOV	127
TAPEIN	191	VARRTN	55
Tape Input File Options (Phase 4)	171	VB1	44
Tape Input Routine (Phase 1)	75	VL1	43
TAPEO (FLR)	145	VLESSG	64
TAPEO (VLR)	163	VLRINZ (ADDROUT Run)	82
Tape Output File Options (Phase 4)	170	VLRINZ (Sort Run)	73
TAPEX	178		
TAPLLR (FLR)	134	WAY4 (FLR)	99
TAPLLR (VLR)	149	WAY4 (VLR)	115
TAPOFX	174	WLR	179
TAPOFX+8	174	WLR2	179
TAPOUT (DSORT401)	169	WLTBLK	162
TAPOUT (DSORT402)	187	WRITE (FLR)	109
TAPPUT	186	WRITE (VLR)	127
TEST	178	WRITEV	185
TESTDEV	46	WTCKMG	66
TESTEQ (FLR)	133	WTCKPS	67
TESTEQ (VLR)	149	WTDATA (FLR)	109
TESTEX	173	WTDATA (VLR)	127
TESTON	173		
TINBG	75	X31CLS (FLR)	148
TMEOF	178	X31CLS (VLR)	165
TMINUS	115	X31CLS-12 (FLR)	148
TPOUT	175	X31CLS-12 (VLR)	165
TR1	177	X31EVT (FLR)	148
Track Capacity, Use of		X31EVT (VLR)	165
Phase 3	160	X31LNK (FLR)	147
Phase 4	184	X31LNK (VLR)	164
TSBPASS	173	X31TS1 (FLR)	147
TSER7	49	X31TS1 (VLR)	165
TSMERG	66		
TSREST	66	ZNLST2	198
TSTBLKCD	26	Zoned Decimal (SIGZON) Convert/Reconvert	
TSTCALCS	65	Routine	197
TSTER5	42	ZONPS2	198
TSTFORM	172	ZYXWZY (Phase 2)	109
TSTREC	174	ZYXWZY (Phase 4)	185
		ZYXWZY-INSERT (Phase 3)	145
UPDTO	184		
UPMOVE	28		
USEORIG	52		
USEORIGR	59		

READER'S COMMENT FORM

IBM System/360
Disk Operating System
Sort/Merge

Y24-5021-0

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM Branch Offices.

- | | Yes | No |
|---|---|--------------------------|
| ● Does this publication meet your needs? | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you find the material: | | |
| Easy to read and understand? | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use? | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete? | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated? | <input type="checkbox"/> | <input type="checkbox"/> |
| Written for your technical level? | <input type="checkbox"/> | <input type="checkbox"/> |
| ● What is your occupation? _____ | | |
| ● How do you use this publication? | | |
| As an introduction to the subject? <input type="checkbox"/> | As an instructor in a class? <input type="checkbox"/> | |
| For advanced knowledge of the subject? <input type="checkbox"/> | As a student in a class? <input type="checkbox"/> | |
| For information about operating procedures? <input type="checkbox"/> | As a reference manual? <input type="checkbox"/> | |
| Other _____ | | |
| ● Please give specific page and line references with your comments when appropriate. If you wish a reply, be sure to include your name and address. | | |

COMMENTS:

- Thank you for your cooperation. No postage necessary if mailed in the U. S. A.

Fold

Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES



POSTAGE WILL BE PAID BY . . .

IBM Corporation
P. O. Box 6
Endicott, N. Y. 13760

Attention: Programming Publications, Dept. 157

Fold

Fold

Cut Along Line

IBM S/360

Printed in U. S. A.

Y24-5021-0



International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N. Y. 10601

Additional Comments:





International Business Machines Corporation

Data Processing Division

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