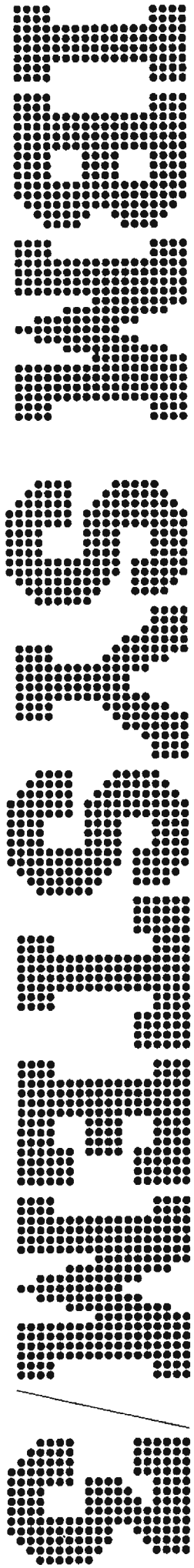


IBM System/3
Disk Sort Program
Logic Manual

Program Numbers:

5703 - SM1 Model 6
5702 - SM1 Model 10 Disk System
5705 - SM1 Model 12
5704 - SM1 Model 15

LY21-0517-6
File No. S3-33

Program Product

Seventh Edition (December 1975)

This is a major revision of, and obsoletes, LY21-0517-5. A vertical line in a left margin indicates a change to the text or a minor change to an illustration. The symbol ● by a caption indicates a major change to an illustration.

This edition applies to version 04, modification 00 of the IBM System/3 Model 15 Disk Sort (Program Product Number 5704-SM1); it also applies to the IBM System/3 Model 6 Disk Sort (Program Product Number 5703-SM1), IBM System/3 Model 10 Disk System Disk Sort (Program Product Number 5702-SM1), and the IBM System/3 Model 12 Disk Sort (Program Product Number 5705-SM1); and to all subsequent versions and modifications until otherwise indicated in new editions or technical newsletters.

Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/3 Bibliography*, GN20-8080, for the editions that are applicable and current.

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

A form for readers' comments is at the back of this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901.

This program logic manual is designed to satisfy requirements of support personnel responsible for maintenance of the IBM System/3 Disk Sort program, whether it is used with the IBM System/3 Model 6, the IBM System/3 Model 8, the IBM System/3 Model 10 Disk System, the IBM System/3 Model 12, or the IBM System/3 Model 15. Information in this manual includes general data and logic flow of the program and detailed descriptions of the program phases, data areas, and object program.

This manual refers to the 5444 Disk Storage Drive, the 5445 Disk Storage, and the 3340 Direct Access Storage Facility. The disk storage device attached to the system determines the meaning of the references. The following tables will assist the user in determining the meaning of the reference(s):

For Systems *without* 3340 Direct Access Storage Facility

Reference	Meaning
5444	5444 Disk Storage Drive
5445	5445 Disk Storage
3340	Not applicable

For Systems *with* 3340 Direct Access Storage Facility

Reference	Meaning
5444	5444 simulation area on 3340 data module
5445	Main data area on 3340 data module
3340	Main data area on 3340 data module

For ease of illustration, many of the examples in this book use card-like figures to represent records. This does not imply that a card device must be used for input or output in these situations. Any of several input/output devices might be used, depending on which System/3 model and configuration you are using.

This publication is intended to be a recall mechanism and a debugging tool. In debugging, however, this manual best serves as a guide to the functional sequence of the instructions in the program listing.

The System/3 Model 8 is supported by System/3 Model 10 Disk System Control Programming and Program Products. The facilities described in this publication for the Model 10 are also applicable to the Model 8, although the Model 8 is not referenced.

RELATED PUBLICATIONS

The following IBM System/3 reference manuals are recommended for additional information:

Model 6

- *IBM System/3 Disk Sort Reference Manual*, SC21-7522.
- *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual*, GC21-7516.
- *IBM System/3 Model 6 Components Reference Manual*, GA34-0001.
- *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual*, SY21-0512.
- *IBM System/3 Disk Systems System Control Program Logic Manual*, SY21-0502.

Model 8

- *IBM System/3 Model 8 Introduction*, GC21-5114.

Model 10 Disk System

- *IBM System/3 Disk Sort Reference Manual*, SC21-7522.
- *IBM System/3 Model 10 Disk System Control Programming Reference Manual*, GC21-7512.
- *IBM System/3 Model 10 Components Reference Manual*, GA21-9103.
- *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual*, SY21-0512.
- *IBM System/3 Disk Systems System Control Program Logic Manual*, SY21-0502.

Model 12

The availability date for the following Model 12 manuals is not the same as for this manual. Orders sent shortly after the edition date of this manual may be considered invalid.

- *IBM System/3 Model 12 System Control Program Logic Manual*, SY21-0046.
- *IBM System/3 Model 12 Introduction*, GC21-5116.
- *IBM System/3 Model 12 System Control Programming Reference Manual*, GC21-5130.
- *IBM System/3 Model 8, 10, 12, and 15 Components Reference Manual*, GC21-9236.

Model 15

- *IBM System/3 Disk Sort Reference Manual*, SC21-7522.
- *IBM System/3 Model 15 System Control Programming Reference Manual*, GC21-5077.
- *IBM System/3 Model 15 Components Reference Manual*, GA21-9193.
- *IBM System/3 Model 15 Supervisor and IOS Logic Manual*, SY21-0033.
- *IBM System/3 Model 15 Data Management Logic Manual*, SY21-0034.
- *IBM System/3 Model 15 Scheduler Logic Manual*, SY21-0035.

Program Logic Manuals

Program logic manuals are also referenced in this manual. In the flowcharts, these references are made indirectly by using function/module names in library blocks (see *Appendix B. Flowcharting Techniques* for an example). The following chart shows which program logic manual to refer to for a discussion of each function/module name used in the flowcharts:

Model 6 and Model 10 Disk System	Function/Module Name						
	CLOSE	OPEN	DM	IOS	SUPV	SYSLOG	EOJ
IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual, SY21-0512.	X	X	X	X			
IBM System/3 Disk Systems System Control Program Logic Manual, SY21-0502.					X	X	X
Model 15							
IBM System/3 Model 15 Supervisor and IOS Logic Manual, SY21-0033.				X	X		
IBM System/3 Model 15 Data Management Logic Manual, SY21-0034.	X	X	X				
IBM System/3 Model 15 Scheduler Logic Manual, SY21-0035.						X	X
Model 12							
IBM System/3 Model 12 System Control Program Logic Manual.	X	X	X	X	X	X	X

Note: The *IBM System/3 Model 15 System Data Areas and Diagnostic Aids Handbook*, SY21-0032, contains all system data area formats.

HOW THIS PUBLICATION IS ORGANIZED

This publication is divided into eight sections:

1. *Introduction* contains general information about the functions and characteristics of the program.

2. *Method of Operation* describes the data flow and logic flow of the program in general terms.

3. *Program Organization* describes the organization of each phase and routine using narrative, flowcharts, and diagrams. Flowcharts are designed to provide easy reference to the program listings.

4. *Directory* contains information needed for quick reference to the program listings.

5. *Data Area Formats* describes the significant data areas used by the programs.

6. *Object Program* describes the Select/Build routine and the generated code which makes up the routine.

7. *Appendix A* discusses the diagnostic aids built into the program.

8. *Appendix B* discusses the flowcharting techniques used.

The System/3 Disk Sort programs are disk resident programs designed to sort records from a disk or tape file (or from card or 3741 files on Model 15). These programs are capable of performing either an ADDROUT sort or a tag-along sort. The output of an ADDROUT sort is an output file (ADDROUT file) containing 3-byte binary relative record numbers (zero indicates the first record in the file) of the input records. The output of a tag-along sort is sorted output records. These output records may contain all the information in the input records or only selected fields from the input records.

In addition to creating the output file, Disk Sort produces a report on the logging device if requested of the sort specifications. This report consists of a listing of all the source specification statements and diagnostic and status messages. Input to Disk Sort consists of OCL statements, the sort specification statements, and the disk or tape input file.

SYSTEM REQUIREMENTS

The minimum system configuration for Disk Sort is:

- A System/3 processing unit
- A disk storage drive
- A system input device
- A system output device

PROGRAM STRUCTURE

Disk Sort can be divided into two major groups of phases: generation phases and execution phases. A brief discussion of each group follows.

Generation Phases

The Disk Sort generation phases perform these functions:

- Set variables in COMMON to reflect whether 5444, 5445, or tape data management is required (or from card or 3741 files on Model 15).
- Read the sort specification statements.
- Print a source listing with diagnostics if requested.
- Generate the Select/Build routine which determines if an input record is to be included in the sort and if a work record is to be built.
- Design the sort.
- Allocate work file automatically, if work file statement is omitted.
- Handle error conditions.

Execution Phases

The Disk Sort execution phases perform these functions:

- Select the input records to be included in the sort using the Select/Build routine generated by the generation phases.
- Build a work record for each of the input records to be included using the Select/Build routine.
- Sort work records into strings on a work file.
- Merge strings in ascending or descending sequence.
- Place the sorted work records on the output file.

This section describes the general functional flow of logic and data through the generation phases and the execution phases of Disk Sort. Diagrams illustrate the flow of logic and data.

COMMON COMMUNICATION REGION (MODEL 6 AND MODEL 10 DISK SYSTEM) AND COMMON COMMUNICATION AREA (MODEL 12 AND MODEL 15)

The first 256 bytes of storage allocated to Disk Sort contain the common communication region (COMMON) for Model 6 and Model 10 Disk System or the common communication area (COMMON) for Model 12 and Model 15. COMMON contains information used by both the generation and the execution phases of Disk Sort. See *Section 5. Data Area Formats* for a complete description of the format and contents of COMMON.

GENERATION PHASES

Figure 2-1 shows the overall data and logic flow through the generation portion of Disk Sort. For a more detailed description of the generation phases and routines, see *Section 3. Program Organization.*

EXECUTION PHASES

The execution portion of Disk Sort is entered upon successful completion of the generation phases. Disk Sort execution consists of two major sections:

1. Execution phases
2. Select/Build routine

The execution phases read records, merge and sort selected records, and place the sorted records on the output file. The Select/Build routine is used by Phase 1A, 1B, or 1X to determine if a record read from the input file should be included or omitted for the sort and to build the work record if the record is included. For a detailed description of the Select/Build routine, see *Section 6. Object Program (Select/Build Routine).*

Figure 2-2 shows the overall data and logic flow through the execution portion of Disk Sort. For a more detailed description of the execution phases and routines, see *Section 3. Program Organization.*

Although more than one symbol for an input or work file is shown in the overall data and logic flowchart (Figure 2-2), there is actually only one of either of these files for a Model 6 or a Model 10. A Model 12 or Model 15 user can have multiple input files. More than one file of a type is shown to indicate the interaction which takes place.

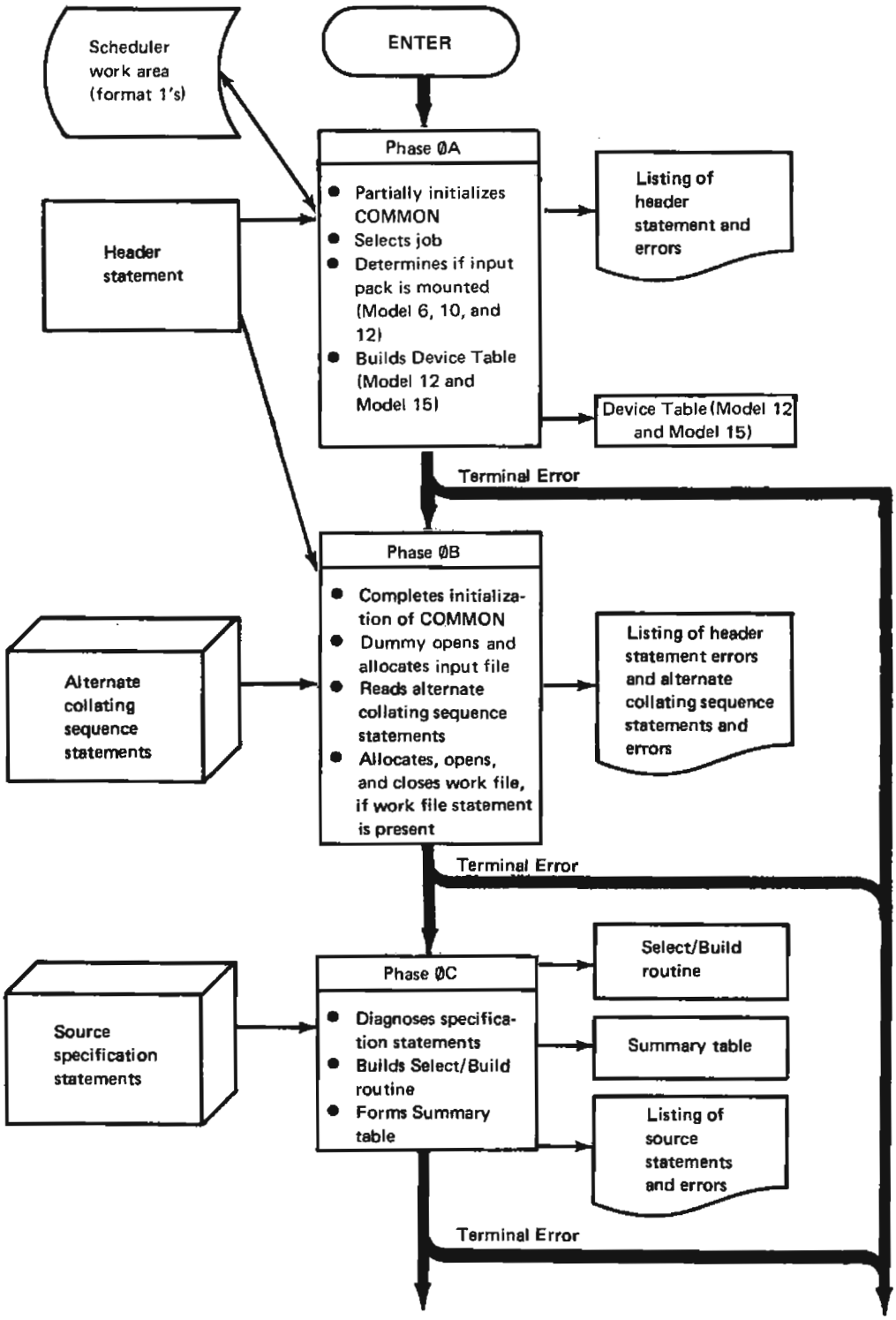


Figure 2-1 (Part 1 of 2). Logic and Data Flow for Sort Generation Phases

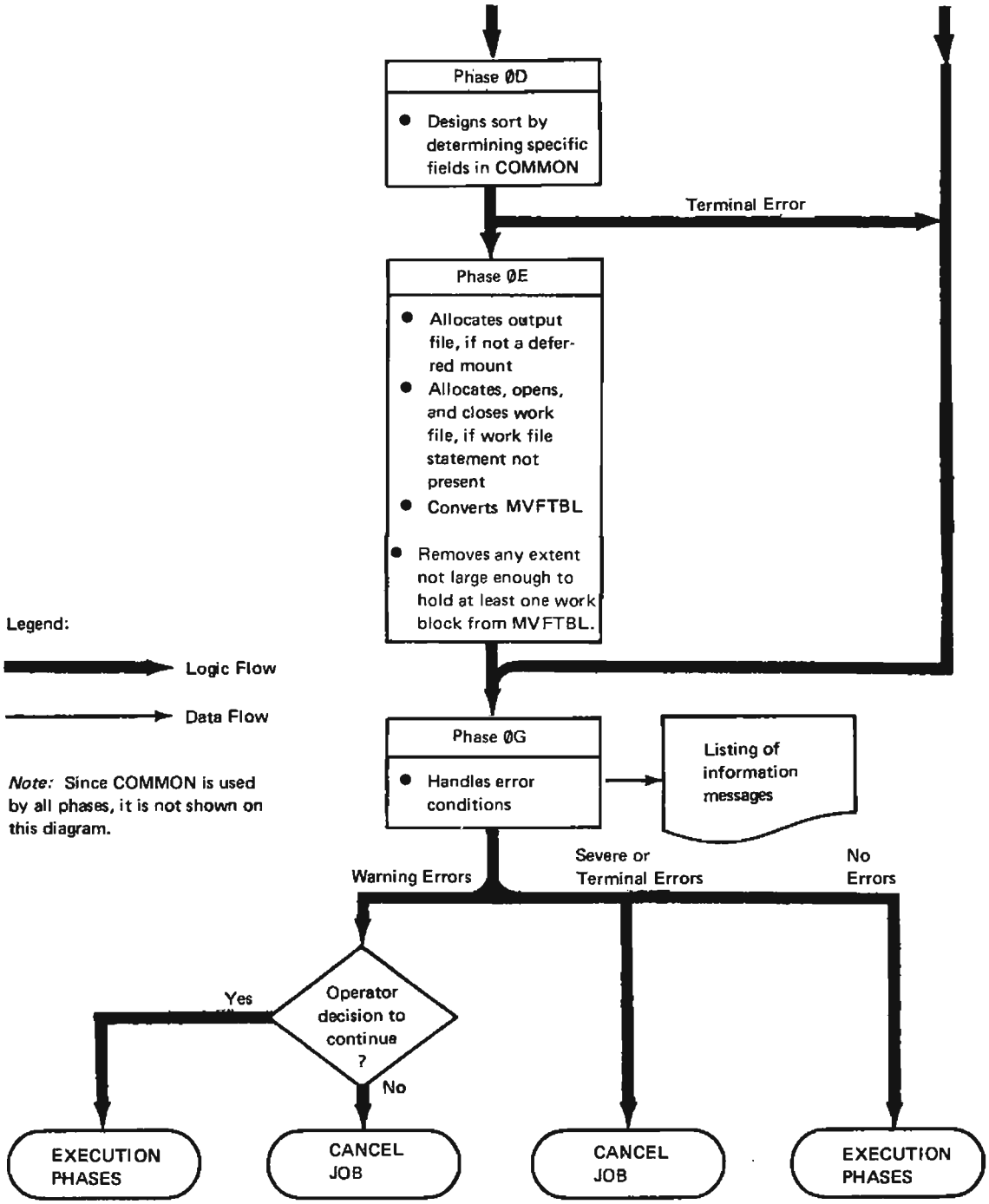
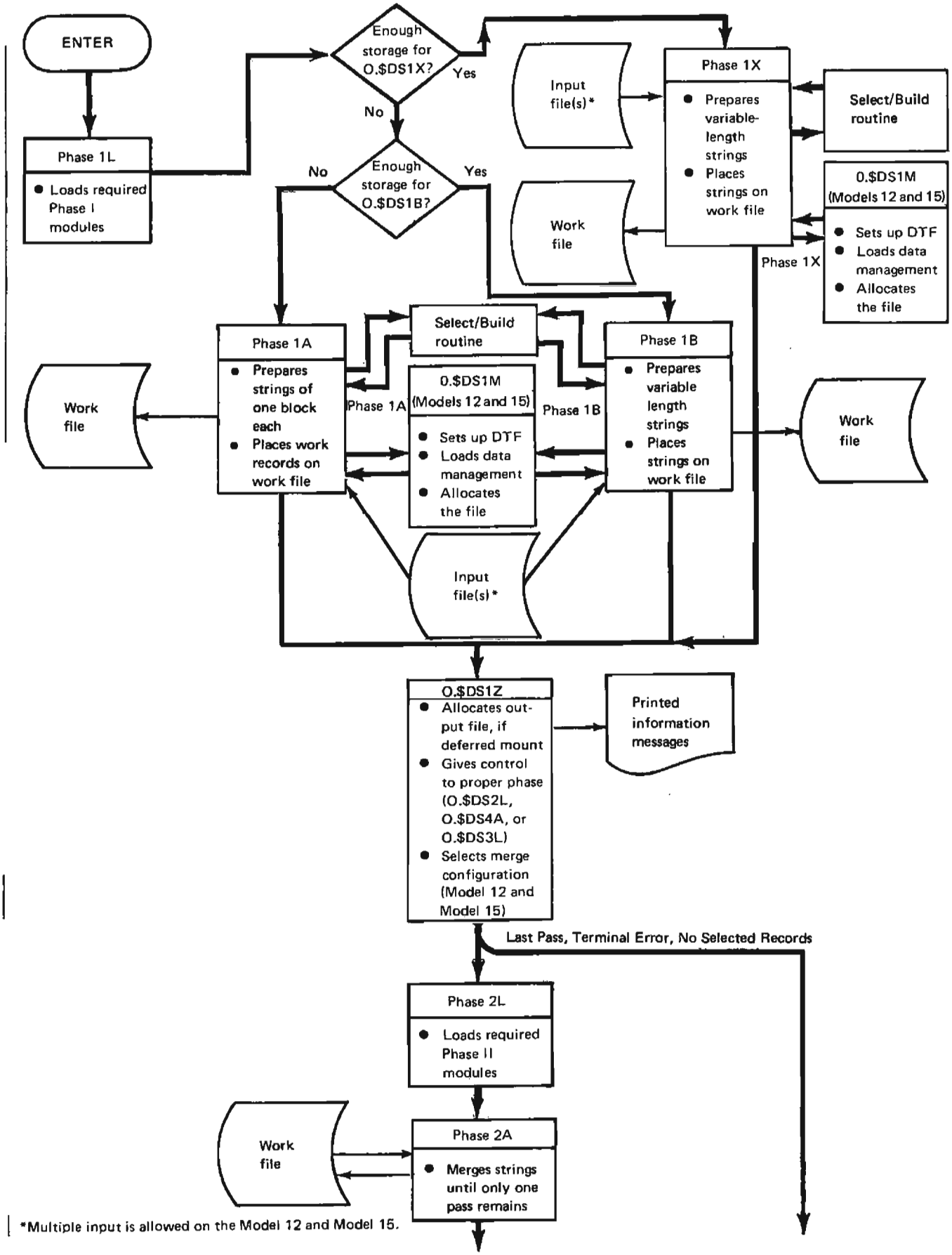


Figure 2-1 (Part 2 of 2). Logic and Data Flow for Sort Generation Phases



*Multiple input is allowed on the Model 12 and Model 15.

Figure 2-2 (Part 1 of 2). Logic and Data Flow for Sort Execution Phases

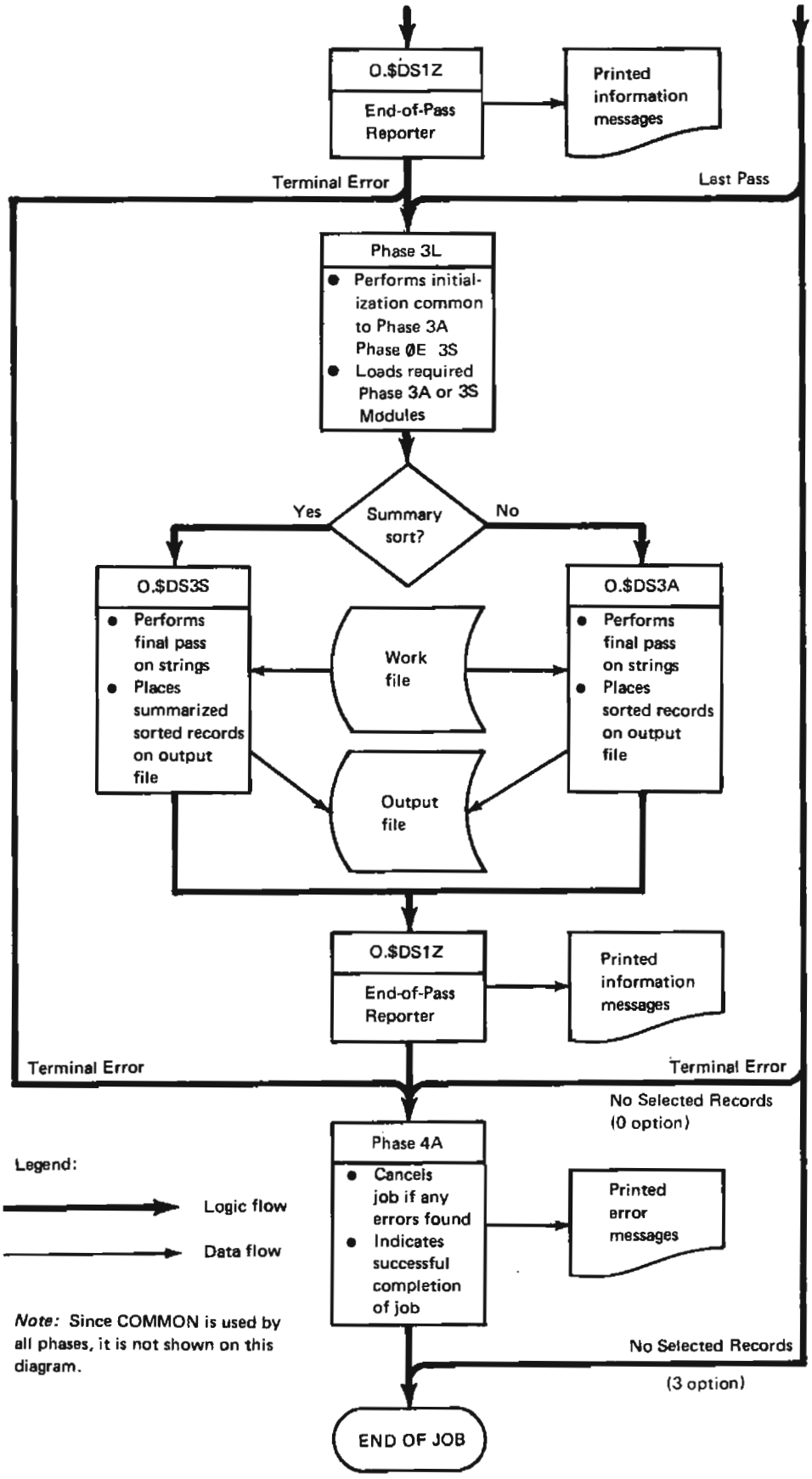


Figure 2-2 (Part 2 of 2). Logic and Data Flow for Sort Execution Phases

This section provides a detailed description of each phase and routine used in the Disk Sort program. A symbolic label used to identify the phase or routine follows each descriptive phase name. Symbolic labels preceded by an "O." indicate a load module. A load module is the phase and/or routines which are loaded into storage as a single unit. The storage maps of the phases indicate the contents of the load module.

Data management routines used are listed but are not discussed in this publication. Refer to:

1. *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual*, SY21-0512, for Model 6 and Model 10 Disk System

2. *IBM System/3 Model 12 System Control Program Logic Manual*, for Model 12

3. *IBM System/3 Model 15 Data Management Logic Manual*, SY21-0034, for Model 15

The SYSIN routine mentioned in the phase descriptions is either the system SYSIN routine, the Scheduler Work Area Get routine, or the Source Library Get routine. $DSAB (Locate Control Statements routine) selects the routine to use. The address of the selected routine is placed in SYSIN@ in COMMON.

Since all phases and most routines use COMMON, references or changes to COMMON are not discussed in this section. See *Section 5. Data Area Formats* for information on COMMON.

GENERATION PHASES

Phase ØA (O.$DSORT)

Phase ØA is divided into three routines:

1. Mainline routine ($DSAA)

2. Locate Control Statements routine ($DSAB)

3. Check for Work File Statement routine ($DSAF)

Mainline Routine ($DSAA)

ENTRY: From Scheduler
STORAGE MAP: Figure 3-1
FUNCTION:
- Records program usage information in COMMON.
- Sets on Rollout bit of NPEOJ in the program communication region (Model 6, Model 10 Disk System and Model 12) or the program communication area (Model 15) in the Supervisor.
- Allocates available storage for the Disk Sort program if the dual programming feature was specified and Disk Sort was loaded in program level 1 (Model 10 Disk System and Model 12).
- Checks for work file statement ($DSAF).
- Loads the address of COMMON in XR1 where it remains for most of the sort.
- Determines if the statement read by $DSAB is a header statement and, if it is, moves the print option entry to COMMON (PRTOPN).
- Determines the type of sort specified on the header statement (SORTA; SORTR, or SORTRS) and sets a bit (ATTRQ1) in COMMON.
- Sets a bit in COMMON (ENVQ2) if DEBUG was specified in positions 2Ø-24 of the header statement so that COMMON is dumped after Phases ØA, ØB, ØD, and all passes.
- Initializes the phase load address in COMMON (PHASE@).

INPUT: Header statement
OUTPUT:
- Printed program title and heading lines.
- Printed listing of the header statement.
- Printed error messages if errors were detected.
- Address of COMMON in XR1.

ROUTINES USED: $DSAB, $DSAF
EXIT: O.$DSBA

Locate Control Statements Routine ($DSAB)

ENTRY: From $DSAA
STORAGE MAP: Figure 3-1
FUNCTION:

- Determines where the sort specification statements are. If the statements are in the system SYSIN device, O.$DS9I is loaded; if the statements are in the Scheduler Work Area, O.$DS9W is loaded; if the first statement was a //SOURCE statement, O.$DS9S is loaded.
- Issues a programmed halt if the source member name cannot be found.
- Determines the length of the loaded module and places the value in SYSINL in COMMON; sets SYSIN entry point in SYSIN@ in COMMON.
- Reads first sequence specifications statement.

INPUT:

- Address of COMMON in XR1.
- Source statement.
- Access to NPSCH2 in the program level communication area to determine where the source statements are.

OUTPUT:

- Header statement (if it is read).
- Halt display ('21' on Model 10 Disk System, Model 12, and Model 15; 'CD235' on Model 6) if source member name was not found.

ROUTINES USED: O.$DS91, O.$DS9W, O.$DS9S
EXIT: $DSAA

Check for Work File Statement Routine ($DSAF)

ENTRY: From $DSAA
STORAGE MAP: Figure 3-1
FUNCTION:

- Sets scheduler interlock on for the program level (Model 10 Disk System and Model 12) or enqueue scheduler interlock for the partition (Model 15) Disk Sort is using.
- Reads in all format 1 records in the scheduler work area for Disk Sort to determine if there is a work file statement.
- Sets off deferred mount bit in each format 1 record (F1ATT, bit 2) so that automatic work file allocation requests a mount if the output is not online.
- Writes the format 1's back to the scheduler work area.
- Determines if the input file is online (not on Model 15); if it is not, $DSAF prints a message and passes control to O.$DSGA.
- Sets bits on in FILE1 in COMMON to represent device type for the input, work, and output files.
- Sets bits on in DATA1 in COMMON to indicate if the 5445 Disk Sort feature (Model 10 Disk System only) is available, and provides 7-track tape information.
- Sets bits on in DATA2 (Model 10 Disk System) or ENVQ1 (Model 12 and Model 15) in COMMON if 5445 disks are available for automatic work file allocation.

- Sets the following indicators in COMMON: ENVQ2 if there is a work file statement; ENVQ3 if R1, R2, or F2 is not available for work space and if output is on a lower drive; ATTRQ1 if the output is a deferred mount.
- Builds the Device Table (Model 12 and Model 15)

INPUT:
- The original format 1's for Disk Sort.
- Address of COMMON in XR1.
- Address of system communication region (Model 6, Model 10 Disk System and Model 12) or system communication area (Model 15).

OUTPUT:
- Updated format 1's for Disk Sort.
- ATTRQ1, ENVQ2, FILE1, DATA1, ENVQ3, and DATA2 (Model 6 and Model 10 Disk System) or ENVQ1 (Model 12 and Model 15).
- Message SD105 if input file is not online (not on Model 15).
- Message SD106 if the work file is on tape.

ROUTINES USED: Scheduler Work Area Read/Write routine

EXIT: Phase $DSAA

Phase ØB (O.$DSBA)

ENTRY: From O.$DSORT

STORAGE MAP: Figure 3-2

FUNCTION:
- Checks for control field length; if not specified, control is passed to O.$DSGA.
- Checks for an invalid number in the control field portion of the header statement ($DSZB); if found, control is passed to O.$DSGA.
- Determines input record length (by using a dummy open of the input file); if indeterminable, control is passed to O.$DSGA.
- Allocates the input file.
- Dummy opens the input file if it is a tape file.
- Allocates the output file if it is a tape file.
- Determines if an ascending or descending sort has been specified. If no sequence has been specified, the program defaults to an ascending sort.

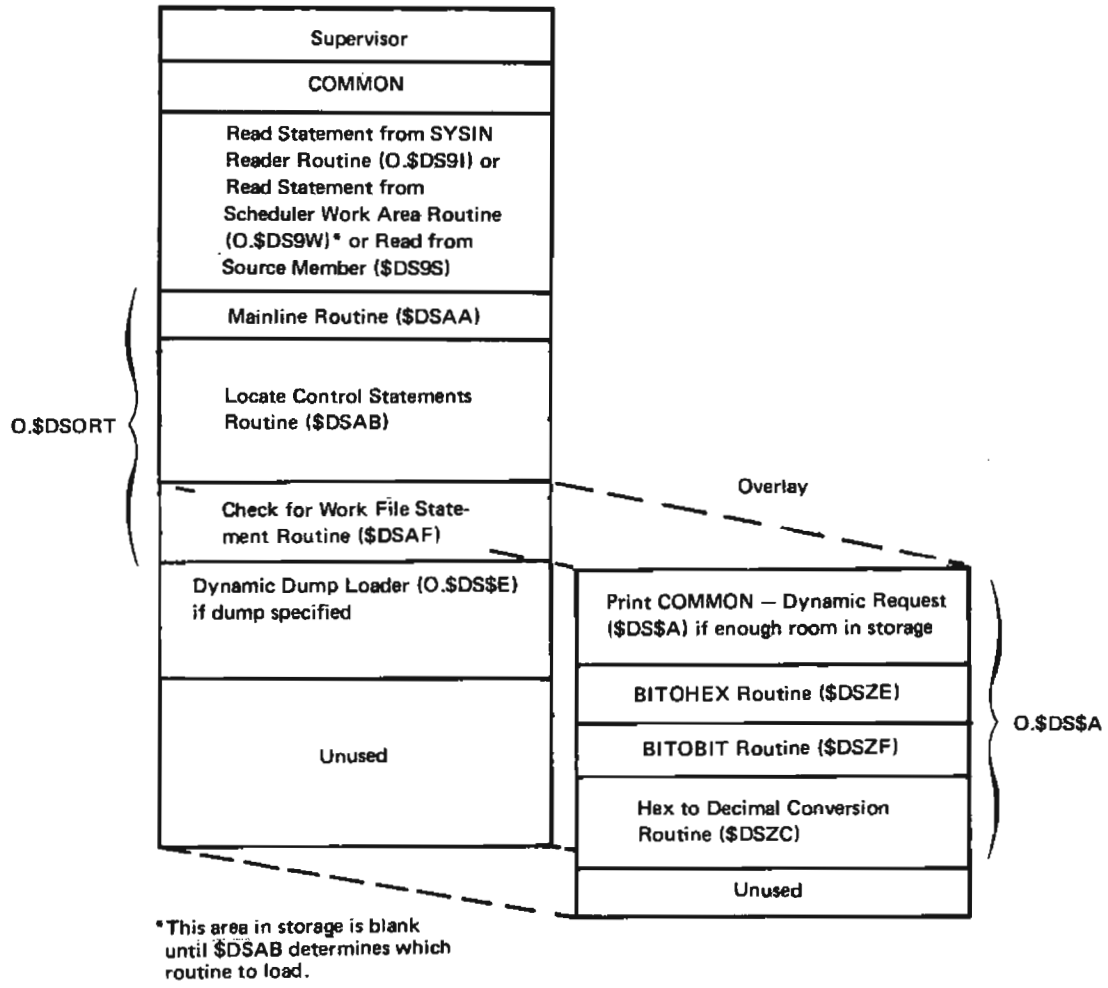


Figure 3-1. Storage Map for Phase ØA

- For a SORTA job:
  1. Sets the work record length equal to the key length plus 3 (WRECL in COMMON).
  2. Sets the output record length (ORECLH in COMMON) to 3 (output tag only).
  3. Sets a bit in COMMON (ATTRQ2) to indicate that the key is to be dropped for this sort.
  4. Ensures that output record length for tape is the same on both sort header and output file statements.
  5. Issues a terminal error if ASCII translate has been specified for an ADDROUT output file on tape.
  6. Issues a terminal error if 7-track tape is output.
- For a SORTR or SORTRS job:
  1. Sets the output record length (ORECLH in COMMON) equal to the length specified on the header statement.
  2. Checks for invalid number in output record length; if found, control is passed to O.$DSGA.
  3. Initially sets the work record length (WRECL in COMMON) equal to the output record length (ORECLH).
  4. Sets WRECL equal to the initial value of work record length plus the key length (KEY + 1) if the output control field is to be dropped.
  5. Issues a terminal error if header and tape output record length are not the same.
  6. Issues a terminal error if 7-track tape files (input and/or output) were specified, and converter or translator was not specified.
- Determines the block length, record length, and fixed block length for the input and output tape file.
- Determines if ASCII is to be used for input and output tape file.
- If tape is used, checks the fields in the input and output DTFs to ensure their agreement.
- Allocates, opens, and closes the work file if a work file statement is supplied.
- Builds the table of work file extents in COMMON (MVFTBL) for the work file.
- Loads and calls O.$DSBC if an alternate collating sequence was specified.
- Sets the alternate collating sequence length field (ALTSEQ in COMMON) to zero if no alternate collating is desired.
- Reads the statement following the header statement.
  1. If an alternate collating sequence was not specified, the address of the SYSIN routine is obtained (SYSIN@ in COMMON) and a branch is taken to the SYSIN routine (O.$DS9I, O.$DS9W, or O.$DS9S).
  2. If an alternate collating sequence was specified, O.$DSBC reads in the statement following the last alternate collating sequence statement (that is, **).
- Computes the Select/Build routine address (BUILD@) and the input record address (IREC@) if no terminal errors are encountered.

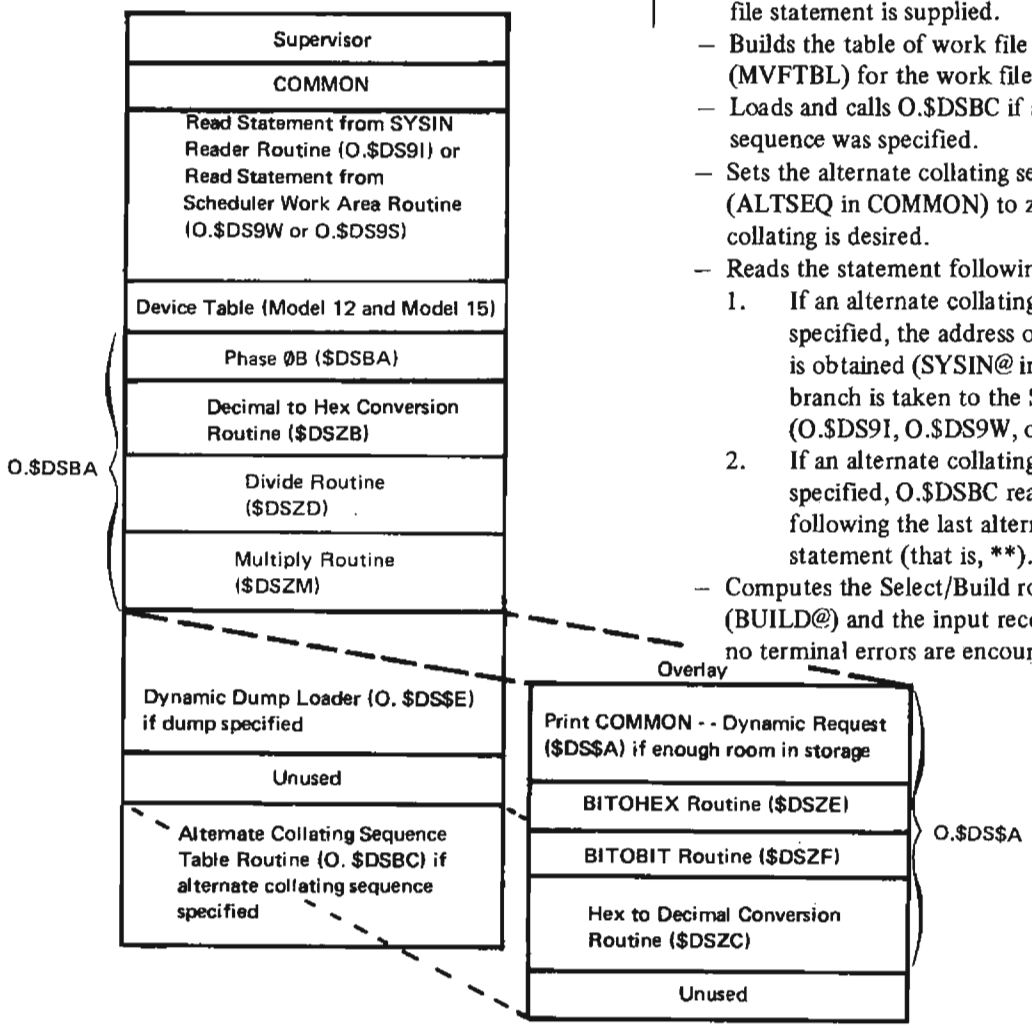


Figure 3-2. Storage Map for Phase ØB

INPUT:
- Address of COMMON in XR1.
- Header statement.

OUTPUT:
- Printed error messages if errors were detected.
- Printed listing of alternate collating sequence statements.
- Table of work file extents (see *Section 5. Data Area Formats* for a table description).
- Bits in COMMON (FILE2) set to designate use of ASCII files.

ROUTINES USED: O.$DSBC, $DSBC, O.$DS9I, O.$DS9W, O.SDS9S, $DSZB, $DSZM, $DSZD
EXIT: O.$DSCA

Phase ØC (Select/Build Routine Compiler)

Phase ØC has three main parts (Figure 3-3), each being a Disk Sort load module. The three parts are:

1. Compile the Select/Build Routine (O.$DSCA)
2. Error Print (O.$DSCB)
3. Move Generation Code (O.$DSCC)

Compile the Select/Build Routine (O.$DSCA)

This part of Phase ØC has nine routines:

1. Initialization routine ($DSCA1)
2. Mainline routine ($DSCA2)
3. Code Segment Move routine ($DSCA5)
4. Determine Zone routine ($DSCA6)
5. Error Table Build routine ($DSCA8)
6. Include/Omit Generator routine ($DSCE)
7. Field Generator routine ($DSCF)
8. Calculate Length Module routine ($DSCL)
9. End of File for Compiler routine ($DSCZ)

Figure 3-4 shows a general overview of these routines. For a more detailed discussion of the code generated by the Select/Build Routine compiler, see *Section 6. Object Program (Select/Build Routine).*

Initialization Routine ($DSCA1)

ENTRY: From O.$DSBA
STORAGE MAP: Figure 3-3
FUNCTION:
- Assures that the statement in the SYSIN area is a non-comment statement (and a non-data statement or non-summary data if SORTA is specified). The statement, whose address is in CARD@ in COMMON, is the first *next* statement used by $DSCA2.
- Prints all bypassed comments or data statements (SORTA only) if the print option entry is blank or Ø.
- Initializes the COMMON fields used by Phase ØC (see COMMON in *Section 5. Data Area Formats*).
- Determines and sets up the linkage for O.$DSCE, O.$DSCF, O.$DSCL, and O.$DSCZ.

INPUT:
- Sequence specification statement image in the input area in the SYSIN routine (CARD@ in COMMON contains its address).
- Address of the SYSIN routine (O.$DS9I, O.$DS9S, or O.$DS9W).
- Address of COMMON in XR1.

OUTPUT:
- Printed listing of all statements and the first non-comment statement.
- Non-comment statement in the read-in area of the SYSIN routine.

ROUTINES USED: O.$DS9I, O.$DS9W, O.$DS9S
EXIT: $DSCA2

Mainline Routine ($DSCA2)

ENTRY: From $DSCA1
STORAGE MAP: Figure 3-3
FUNCTION:
- Fills in the final INO branch when the old *next* statement is an end-of-file statement. O.$DSCZ is then loaded and called.
- Moves the *next* statement, whose address is in CARD@ in COMMON, to the 39-byte area (CARD) in COMMON. This statement is now the new *current* statement.
- Prints the *current* statement if the print option entry is Ø.
- Reads and prints statements until a non-comment statement is found (including end-of-file). This non-comment statement becomes the new *next* statement.
- Checks for valid combinations of positions 6-8 to determine if the statement is a valid record type or field specification; if invalid, control is passed to $DSCA8.
- Calls O.$DSCL unless current statement is an implicit include-all.
- Determines if the specification statement is an include/omit and, if it is, calls O.$DSCE.

INPUT:
- Sequence specification statement image in the read-in area of the SYSIN routine.
- Permanent address of the Select/Build routine.
- Address of the SYSIN routine (O.$DS9I, O.$DS9S, or O.$DS9W).
- Address of COMMON in XR1.

OUTPUT:
- Generated code in the Select/Build routine unless the error table begins to overlay the Select/Build routine. If this occurs, the job is canceled in Phase ØG.
- Printed listing of the source specification statements.
- Error table (see *Section 5. Data Area Formats*).
- Length of the Select/Build routine.
- Length and number of summary table entries (SORTRS).

ROUTINES USED: O.$DSCE, O.$DSCF, O.$DSCL, O.$DS9I, O.$DS9S, O.$DS9W
EXIT: O.$DSCZ

Code Segment Move Routine ($DSCA5)

ENTRY: From $DSCA2, O.$DSCE, or O.$DSCF
STORAGE MAP: Figure 3-3
FUNCTION:
- Moves code generated in O.$DSCE and O.$DSCF to the generated code area.
- Maintains the location counters for the generated code and the temporary select/build area.
- Moves the constants generated in $DSCA9 to the generated code area.
- Determines if there is enough storage available for code to be generated; if not, bit 7 of ATTRQ3 is set in COMMON.

INPUT:
- Address of the length of the code to be moved.
- Address of the displacement of the code in the constant area.
- Address of COMMON in XR1.

OUTPUT:
- Generated code moved to the current place in the temporary select/build area.
- Updated location counters for the length of the code moved and for the temporary and permanent select/build area.
- Updated error table.

ROUTINES USED: $DSCA8
EXIT: The next sequential instruction + 2 of $DSCA2, O.$DSCE, or O.$DSCF

Determine Zone Routine ($DSCA6)

ENTRY: From O.$DSCE or O.$DSCF
STORAGE MAP: Figure 3-3
FUNCTION: Determines the zone of a given byte by testing the zone portion of that byte.
INPUT:
- Byte in CARD@ + 19 in COMMON.
- Address of COMMON in XR1.

OUTPUT: Generated code (code segment 2).
ROUTINES USED: None
EXIT:
- Next sequential instruction of O.$DSCE or O.$DSCF if the entire code segment is not needed.
- Next sequential instruction + 3 of O.$DSCE or O.$DSCF if the entire code segment is needed.

Error Table Build Routine ($DSCA8)

ENTRY: From $DSCAZ, O.$DSCE, O.$DSCF, $DSCA5, $DSCA7, or $DSCA9
STORAGE MAP: Figure 3-3
FUNCTION:
- Builds the error table starting with the highest available program level (partition) address.
- Checks error table; if full, control is passed to O.$DSCZ.

INPUT:
- ARR containing the address of a DC with a 1-byte error number in it.
- Address of COMMON in XR1.
- Error table.

OUTPUT: Updated pointer to the next available space in the error table.
ROUTINES USED: None
EXIT: The next sequential instruction + 1 of $DSCA2, O.$DSCE, O.$DSCF, $DSCA5, $DSCA7, or $DSCA9

Include/Omit Generator Routine (O.$DSCE)

ENTRY: From $DSCA2
STORAGE MAP: Figure 3-3
FUNCTION:
- Diagnoses the specification statements to determine if they are invalid include or omit statements; if not invalid, control is passed to $DSCA8.
- Generates code segments from the information on the include/omit statements.

INPUT:
- Next sequence specification statement in the read-in area of the SYSIN routine.
- First 39 bytes of the *current* statement in the work area in COMMON.
- Address of COMMON in XR1.

OUTPUT:
- Generated code.
- Updated error table.

ROUTINES USED: $DSCA5, $DSCA6, $DSCA8
EXIT: $DSCA2

Field Generator Routine (O.$DSCF)

ENTRY: From $DSCA2
STORAGE MAP: Figure 3-3
FUNCTION:
- Builds summary table entry for summary data field.
- Generates code segments from the information on the field statements ($DSCA5, $DSCA6).
- Builds code to move the three-byte binary relative record number behind the control word for SORTA jobs.
- Passes control to $DSCA8 if field statement errors are found.

INPUT:
- Next sequence specification statement in the read-in area of the SYSIN routine.
- First 39 bytes of the *current* statement in the work area in COMMON.
- Address of COMMON in XR1.

OUTPUT:
- Generated code.
- Updated error table.
- Summary table.

ROUTINES USED: $DSCA5, $DSCA6, $DSCA8
EXIT: $DSCA2

Calculate Length Module (O.$DSCL)

This routine is divided into two routines:

1. Calculate Length Mainline routine ($DSCA9)

2. Decimal to Binary Convert routine ($DSCA7)

Calculate Length Mainline Routine ($DSCA9)

ENTRY: From O.$DSCA2
FUNCTION:
- Moves the From field (positions 9-12) of Factor 1 and the To and From fields (positions 20-27) of Factor 2, if present, to the To field of Factor 1 (positions 13-16). $DSCA7 is then called.
- If the 39-byte work area contains an include or omit statement:
  1. Factor 1 data is used to calculate the read area displacement and the length of the field.
  2. The displacement and length of Factor 2 is calculated if Factor 2 is not a constant.
  3. If Factor 2 is a constant, the following occurs:
     a. The constant is translated if alternate collating sequence is requested.
     b. A jump instruction is moved to the code area.
     c. The zones of the constant are set to Fs if D (digit) is specified in column 8.
     d. Work area displacement is calculated.
- If the 39-byte work area contains a field statement:
  1. The read area displacement and the field length are calculated using the location field (to field, positions 13-16) data.
  2. The control word area displacements calculated previously are set up to be used by the generating routines if the statement being used is a continued force statement.
  3. The control word area displacement is calculated and set up for the generating routines if the statement being processed is not a continued force statement.
  4. Inline code is generated to increment the base register for control word and data statement building.

INPUT:
- 39-byte work area in COMMON (positions 1-32 of the current statement).
- Address of COMMON in XR1.

OUTPUT:
- Calculated length and displacement of Factor 1.
- Calculated length and displacement of Factor 2 if it is not a constant.
- Calculated length and displacement of the field statement specification.
- Updated pointer to the next field in the control statement.

EXIT:
- $DSCA2 if no errors occurred.
- O.$DSCB from $DSCA8 if the error table completely fills the select/build area.

ROUTINES USED: $DSCA5, $DSCA7, $DSCA8, O.$DSBC

Decimal to Binary Convert Routine ($DSCA7)

ENTRY: From $DSCA9
FUNCTION:
- Converts the To field (positions 13-16) of Factor 1.
- Determines if value being converted is larger than input record size.
- Converts the new cumulative length (positions 20-22 of the SORTRS field specifications) to binary.

INPUT:
- Decimal numbers in the 39-byte work area.
- Address of COMMON in XR1.

OUTPUT: Hexadecimal equivalent of the decimal number.
ROUTINES USED: $DSCA8, $DSZB
EXIT:
- $DSCA9 if no errors occurred.
- O.$DSCB from $DSCA8 if the error table completely fills the select/build area.

End of File for Compiler Routine (O.$DSCZ)

ENTRY: From $DSCA2
STORAGE MAP: Figure 3-3
FUNCTION:
- Finalizes Select/Build code.
- Updates phase load address in COMMON (PHASE@).
- Moves the summary table to Phase I location behind COMMON.
- Checks for errors; if errors found, control is passed to $DSCA8.

INPUT:
- Select/Build code.
- Address of COMMON in XR1.

OUTPUT:
- Finalized Select/Build code.
- Summary table moved to Phase I location behind COMMON.
- Updated phase load address in COMMON (PHASE@).

ROUTINES USED: None
EXIT: O.$DSCB

Error Print (O.$DSCB)

ENTRY: From $DSCZ
STORAGE MAP: Figure 3-3
FUNCTION:
- Checks the print option entry in COMMON (PRTOPN) to determine if printing is to be done.
- Prints error messages based on the error table generated in Phase ØC.
- Sets error indicators in COMMON (WRNERQ, SEVERQ, TMLERQ).
- Counts the number of errors and places the updated count in COMMON (WRNER#, SEVER#, TMLER#).
- Determines if any terminal errors were diagnosed from Phase ØC; if they were, control is passed to O.$DSGA.

INPUT:
- Address of the error table in COMMON (BEGER1).
- Address of the next error table entry in COMMON (SEVER1).
- Address of COMMON in XR1.

OUTPUT:
- Printed error messages on the SYSLOG device.
- Updated phase load address in COMMON (PHASE@).

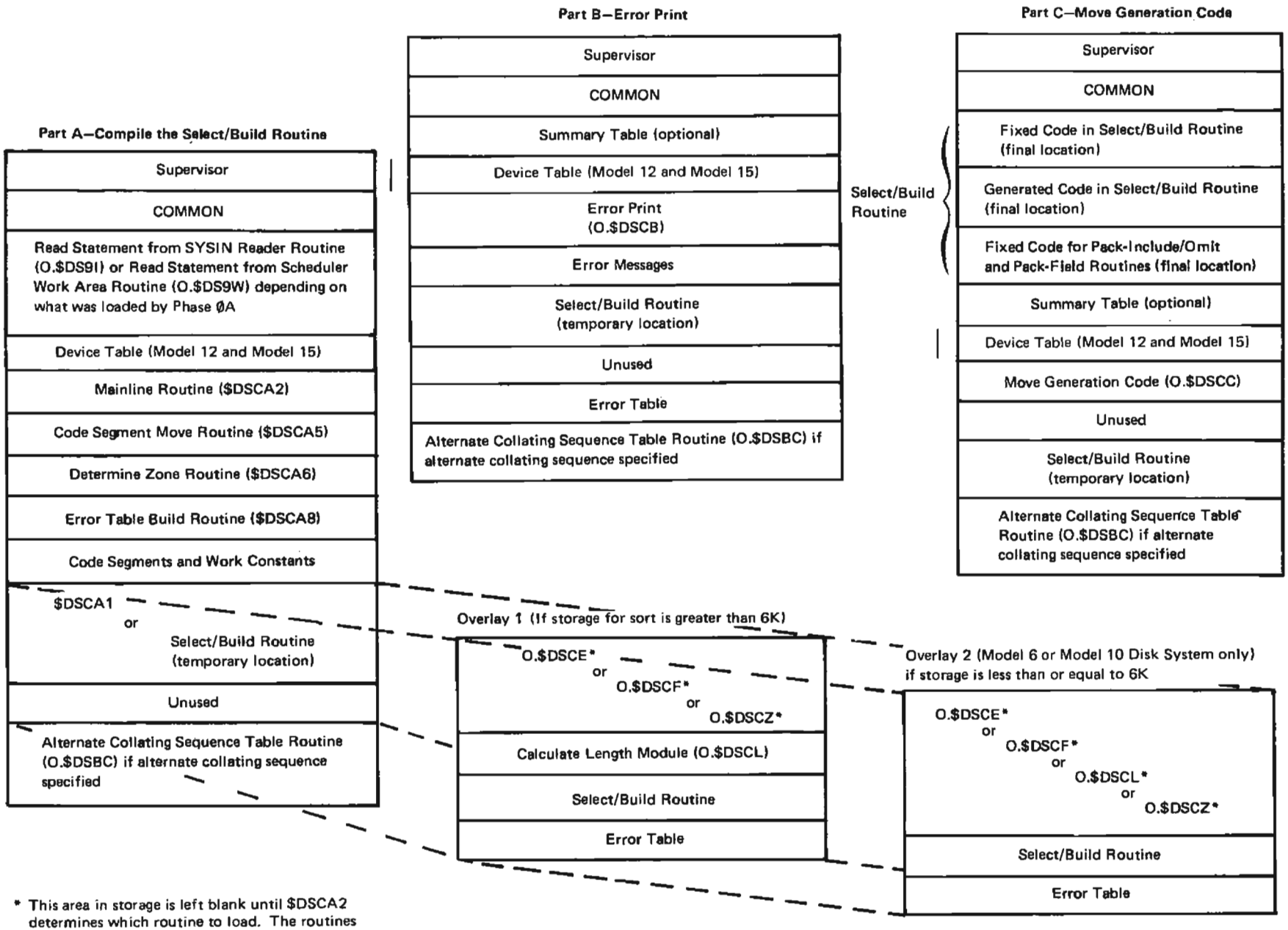
ROUTINES USED: None
EXIT: O.$DSCC

Move Generation Code (O.$DSCC)

ENTRY: From O.$DSCB
STORAGE MAP: Figure 3-3
FUNCTION:
- Moves the code generated by O.$DSCA, O.$DSCE, and O.$DSCF to its final position in storage.
- Moves the fixed code in front of the generated code.
- Moves the fixed code for the Pack-Include/Omit and Pack-Field routines to the end of the Select/Build routine when they are needed.
- Moves the summary table behind Select/Build routine if summary sort is used.

INPUT: Address of COMMON in XR1.
OUTPUT:
- Select/Build routine in final location.
- Summary table behind Select/Build routine (when summary sort used).
- Device Table behind Select/Build routine or behind Summary Table if summary sort is used (Model 12 and Model 15).
- Updated phase load address in COMMON (PHASE@).

ROUTINES USED: None
EXIT: O.$DSDA



* This area in storage is left blank until $DSCA2 determines which routine to load. The routines then overlay one another as many times as necessary.

Figure 3-3. Storage Map for Phase 0C

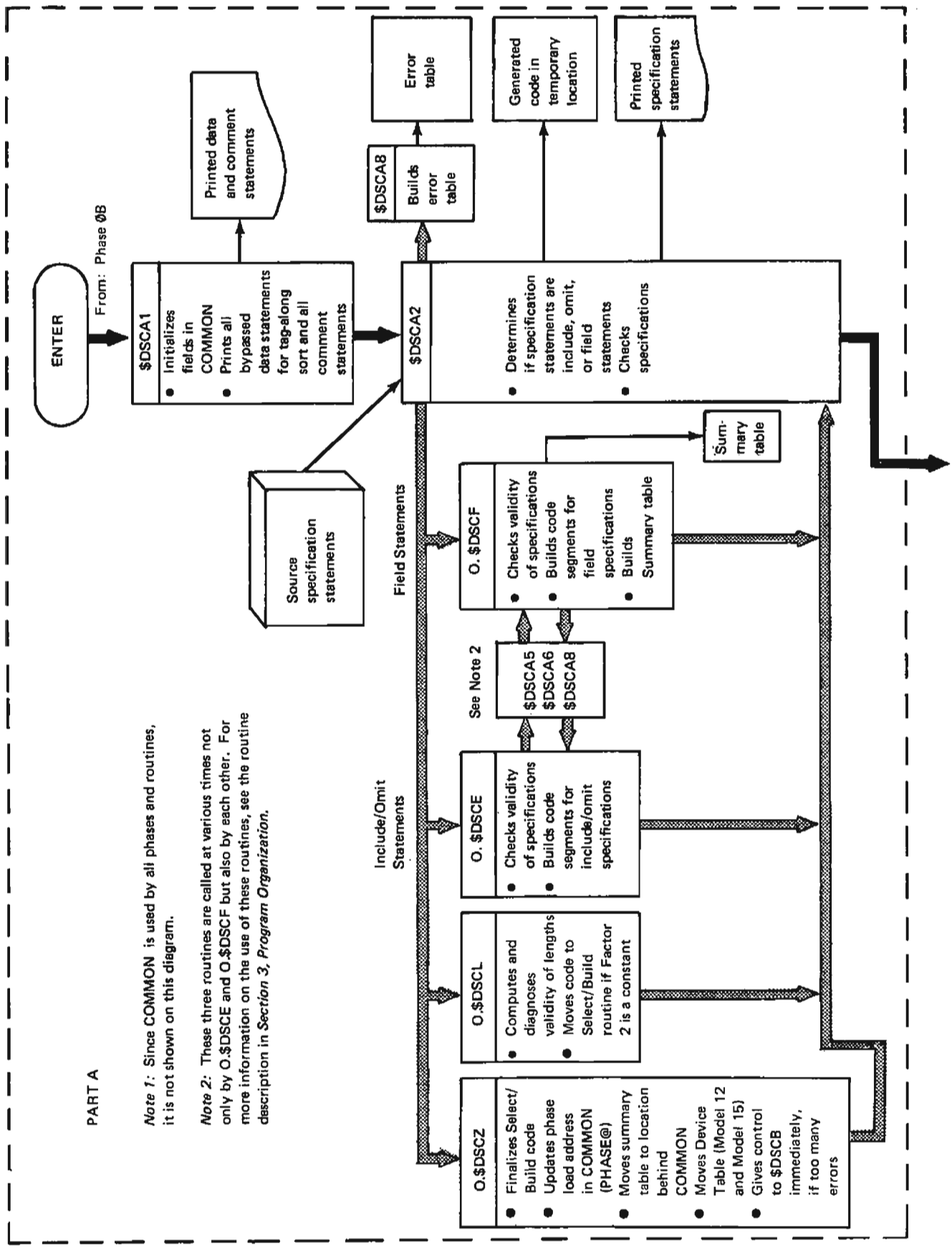


Figure 3-4 (Part 1 of 2). Logic and Data Flow for Phase ØC

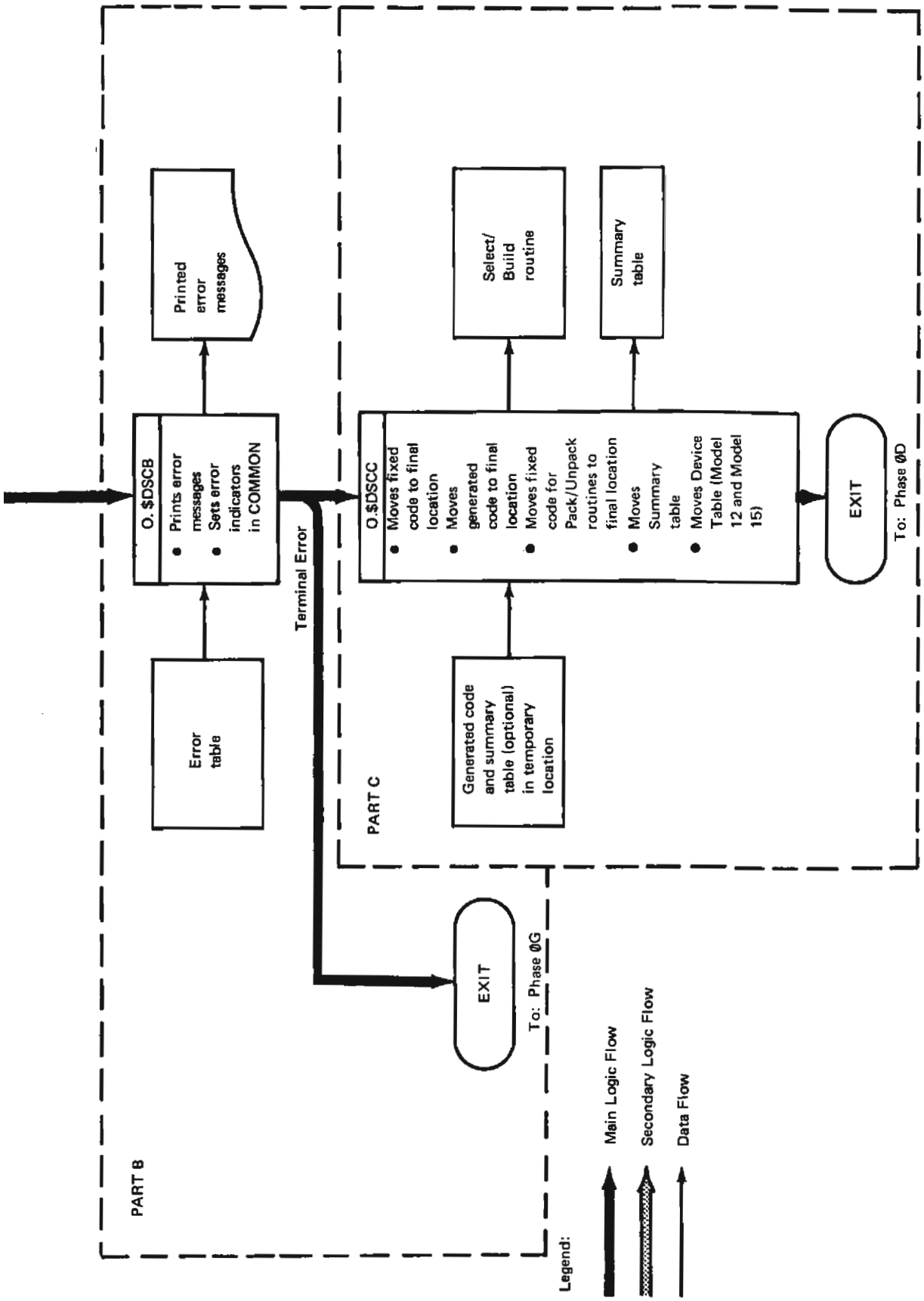


Figure 3-4 (Part 2 of 2). Logic and Data Flow for Phase ØC

Phase ØD -- Part A (O.$DSDA)

ENTRY: From O.$DSCC
STORAGE MAP: Figure 3-5
FUNCTION:

- Computes the active program lengths of Phases 1A, 1B, 1X, 2A, 3A, and 3S.
- Determines if 5445 Disk Storage has been referenced but is not an available feature; if the feature is not available, control is passed to O.$DSGA (Model 10 Disk System).
- Determines the load point for Phase I Data Management (DMLD@) for Model 12 and Model 15.
- Determines maximum number of sectors needed for Phase I Data Management (DMLNG) for Model 12 and Model 15.
- Determines address of the Phase I DTF area (INDTF@) for Model 12 and Model 15.

INPUT:

- Address of COMMON in XR1.
- Active program lengths of execution modules in the load module library.

OUTPUT: The active program lengths of each execution phase (except O.$DS1Z and O.$DS4A).
ROUTINES USED: None
EXIT: O.$DSDB

Phase ØD -- Part B (O.$DSDB)

ENTRY: From O.$DSDA
STORAGE MAP: Figure 3-5
FUNCTION:

- Determines order of merge used in Phase II (OM in COMMON).
- Determines work block length (WBLKL).
- Determines work buffer length (WBUFL).
- Determines work blocking factor (WBF).
- Determines type of internal sort (ISTYPE in COMMON) to be used (Phase 1A, 1B, or 1X).
- Determines input buffer length (IBUFL).
- Determines number of records in the internal sort area (ISA).
- Determines internal sort area length (ISL).
- Determines order of merge for Phase III (OM in COMMON).
- Determines number of output buffers (OBUF#).
- Determines output block length (OBLKL).
- Determines output buffer length (OBUFL).
- Checks for terminal error in design of sort; if found, control is passed to O.$DSGA.

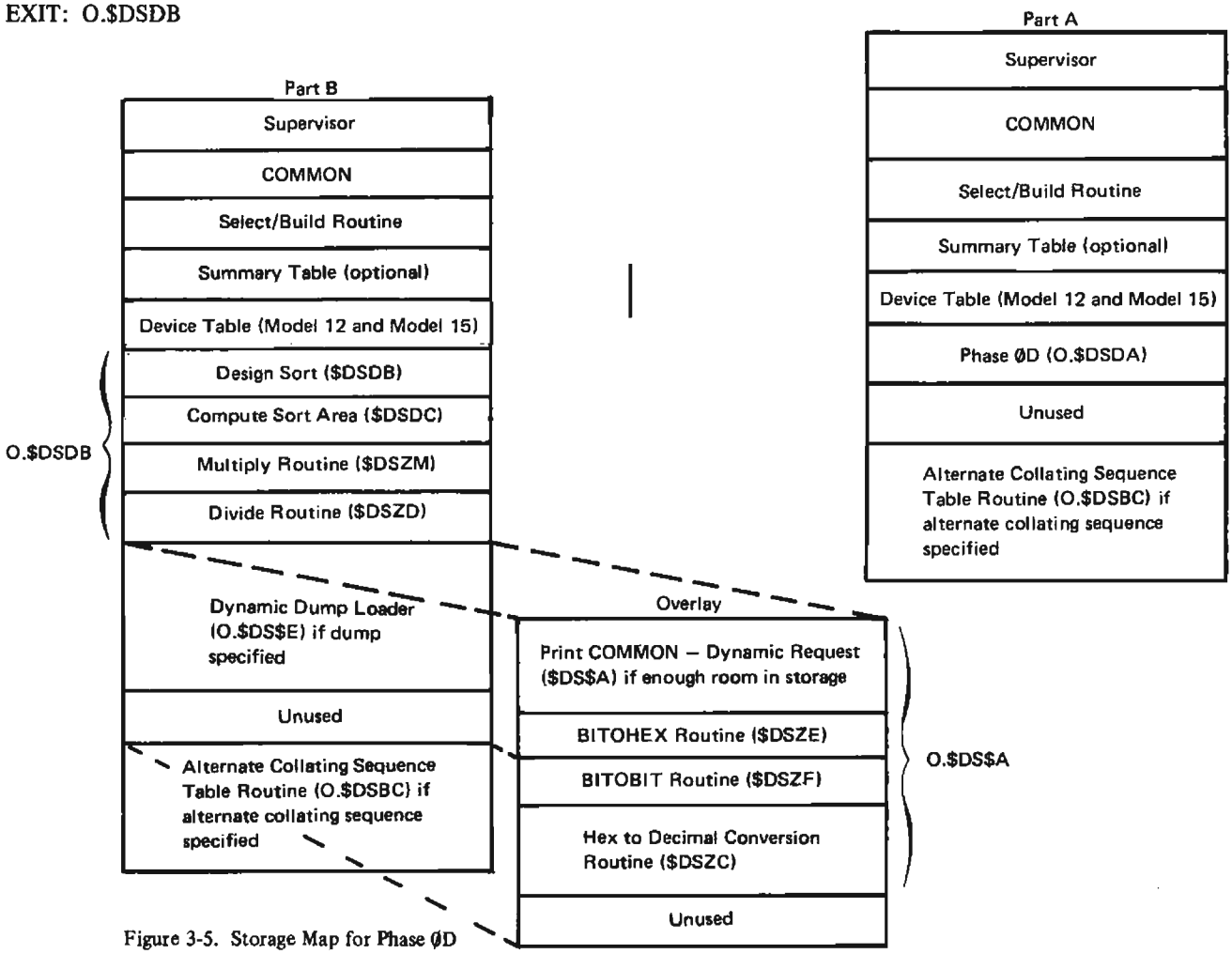


Figure 3-5. Storage Map for Phase ØD

INPUT:
- Active lengths of Phases 1A, 1B, 1X, 2A, 3A, and 3S.
- Address of COMMON in XR1.

OUTPUT:
- Order of merge for Phase II.
- Sort type.
- Order of merge for Phase III.

ROUTINES USED: $DSZD, $DSZM
EXIT: O.$DSEA

Phase ØE (O.$DSEA)

ENTRY: From O.$DSDA
STORAGE MAP: Figure 3-6
FUNCTION:
- Allocates output file if not a deferred mount file.
- Calls automatic allocate ($DSEG) if work file statement omitted.
- Converts number of sectors per entry in table of work file extents to number of blocks per extent.
- Determines if terminal error was found by $DSEG; if error was found, control is passed to O.$DSGA.
- Sorts out any extent in MVFTBL not large enough to hold at least one work block.

INPUT:
- Work file statement indicator (ENVQ2).
- Output deferred mount indicator (ATTRQ1).

OUTPUT:
- Allocated output file if the file is not a deferred mount.
- Automatically allocated work file if there was no work file statement.
- Error indicator (TMLERQ) on if no work space available.
- Updated table of work file extents.

ROUTINES USED: $DSEG
EXIT: O.$DSGA

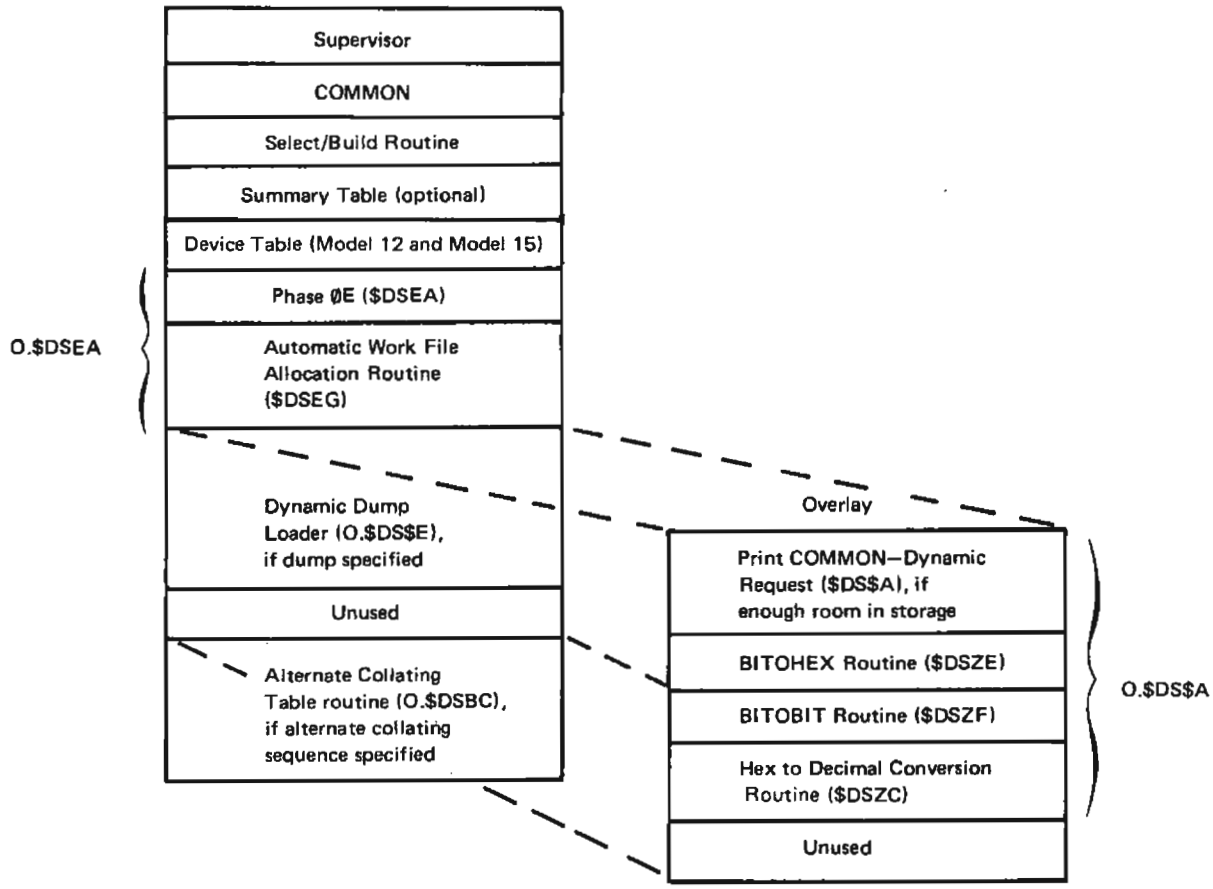


Figure 3-6. Storage Map for Phase ØE

Phase ØG (O.$DSGA)

ENTRY: From O.$DSORT, O.$DSBA, O.$DSCC, O.$DSDA, O.$DSDB, or O.$DSEA
STORAGE MAP: Figure 3-7
FUNCTION:

- Writes a descriptive message indicating permanent I-O errors, severe errors, or terminal errors found in any of the preceding generation phases.
- Issues a programmed halt for I-O errors, terminal errors, or severe errors. The job is then canceled.
- Writes a descriptive message and halts for warning errors. The operator then decides to either continue or cancel the job.
- Writes a message, provided there are no errors and the print option entry is Ø or 1, informing the user that the job has completed the generation portion of Disk Sort.
- Uses $DSZC to convert hexadecimal error warning number to decimal.
- Gives a brief summary of the sort specifications if there were no terminal errors and the print option entry is not 2 or 3.
- If the operator desires to terminate the sort after a halt display of '22' on the Model 10 Disk System, Model 12, and Model 15 or 'C123' on the Model 6, control is passed to EOJ Transient.

INPUT: Address of COMMON in XR1.
OUTPUT:

- Error messages or a message stating that there were no errors in the generation portion of Disk Sort.
- Halt display ('22' on Model 10 Disk System, Model 12, and Model 15; 'C123' on Model 6) for source read warning errors.
- Halt display ('23' on Model 10 Disk System, Model 12, and Model 15; 'CD3' on Model 6) for source read I-O errors.
- Halt display ('25' on Model 10 Disk System, Model 12, and Model 15; 'CD45' on Model 6) for severe or terminal errors.
- Sort specification summary report.

ROUTINES USED: $DSZC
EXIT: O.$DS1L

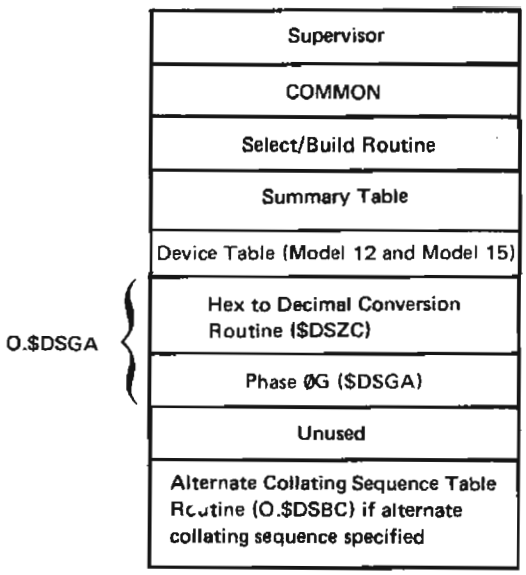


Figure 3-7. Storage Map for Phase ØG

GENERATION ROUTINES

Alternate Collating Sequence Table Routine (O.$DSBC)

This routine has two parts:

1. Table modification
2. Table usage

Note: The last byte of this module must reside in the last byte of the position assigned to O.$DSORT. For this reason, the module must be a multiple of 256. O.$DSBC is padded with an overlay maintenance area to make this possible.

Table Modification Routine

ENTRY: From O.$DSBA
FUNCTIONS:
- Modifies the 256-byte alternate collating sequence table as specified on the alternate collating sequence specification statements that are read in.
- Updates the error counter and indicator for errors found.
- Prints error messages if the print option entry is Ø.
- Determines if the data is valid.
- Reads and prints comments.
- Determines the alternate collating sequence length.

INPUT:
- Address of COMMON in XR1.
- Alternate collating sequence statements.
- Alternate collating sequence table.

OUTPUT:
- Modified alternate collating sequence table.
- Printed error messages if errors were detected.
- Updated error count.
- Next card image in the buffer area within the SYSIN routine.

ROUTINES USED: O.$DS9I, O.$DS9W, O.$DS9S
EXIT: O.$DSBA

Table Usage Routine

ENTRY: From $DSCA9, O.$DS1A, O.$DS1B, or O.$DS1X
FUNCTION: Translates the indicated data and replaces the original data with the translated data.
INPUT:
- Address of COMMON in XR1.
- Address of the area of the statement to be translated (ARR + 1).
- Length of the area to be translated (ARR + 3).

OUTPUT: Translated data in place of the original data.
ROUTINES USED: None
EXIT: $DSCA9, O.$DS1A, O.$DS1B, O.$DS1X

Automatic Work File Allocation Routine ($DSEG)

ENTRY: From O.$DSEA
FUNCTION:
- Allocates work file space.
- Builds table of work file extents for automatically allocated work file.
- Determines if space was actually allocated; if not, control is passed to O.$DSEA.

INPUT: Unit availability indicators (ENVQ3).
OUTPUT: Entries in the table of work file extents.
ROUTINES USED: None
EXIT: Next sequential instruction in O.$DSEA

Decimal to Hexadecimal Conversion Routine ($DSZB)

ENTRY: From O.$DSBA or $DSCA7
FUNCTION:
- Converts an unsigned zoned decimal number (up to seven bytes long) to a 3-byte hexadecimal number.
- Sets ERRQ in COMMON to X'FF' if the zoned decimal number is not valid.

INPUT:
- Address of COMMON in XR1.
- Address of a parameter list in XR2. This address is loaded in XR2 immediately before the branch to the calling routine. This 4-byte list contains:
  1. Address of the 7-byte field in which the zoned decimal number is located.
  2. Address of a 3-byte field where the result is to be placed.

OUTPUT: The 3-byte hexadecimal equivalent of the zoned decimal number in the field specified by the parameter list.
ROUTINES USED: None
EXIT: O.$DSBA or $DSCA7

Hexadecimal to Decimal Conversion Routine ($DSZC)

ENTRY: From O.$DSGA, O.$DS1Z, O.DSA, or O.DSL
FUNCTION:
- Converts a 3-byte hexadecimal number to a 7-byte zoned decimal number and concatenates a sign byte to the result.
- Places the result in the 8-byte field specified by the calling routine. The sign occupies the rightmost byte of the field.

INPUT:
- Address of COMMON in XR1.
- Address of a parameter list in XR2. This address is loaded in XR2 immediately before the branch to the calling routine. The 4-byte list contains:
  1. Address of the 8-byte field in which the resultant zoned decimal number is to be placed.
  2. Address of the 3-byte hexadecimal number to be converted.

OUTPUT: Seven-byte zoned decimal number immediately followed by a sign byte in the field specified in the parameter list.
ROUTINES USED: None
EXIT: O.$DSGA, O.$DS1Z, O.DSA, or O.DSL

Four-Byte Hexadecimal Divide Routine ($DSZD)

ENTRY: From O.$DSBA, O.$DSDB, O.$DS1Z, or O.$DS3L
FUNCTION:
- Divides a 4-byte hexadecimal number into another 4-byte hexadecimal number:
  1. Places the divisor and dividend in a 12-byte work area.
  2. Determines if the divisor is zero.
  3. If the divisor is zero, sets the quotient field and the field originally containing the dividend to X'7FFFFFFF'.
  4. If the divisor is not zero, performs the following:
     a. Executes division by repeated subtraction of the divisor from the dividend.
     b. Gives the correct signs to the resultant quotient and dividend.
     c. Places the quotient in the user-specified quotient field and the remainder in the user-specified remainder field.

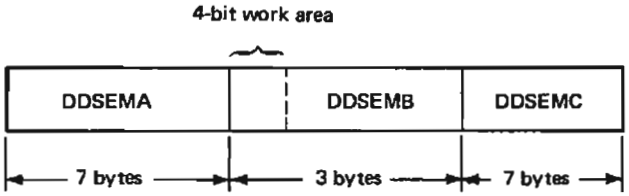
INPUT:
- Address of COMMON in XR1.
- Address of a parameter list in XR2. The list contains:
  1. Address of the four bytes containing the divisor.
  2. Address of the four bytes containing the dividend.
  3. Address of a 4-byte field where the quotient is to be placed.
  4. Address of where the remainder is to be placed.

OUTPUT:
- Resultant quotient in the field specified in the parameter list.
- Remainder in the field specified in the parameter list.

ROUTINES USED: None
EXIT: O.$DSDB, O.$DS1Z, or O.$DS3L

Three-Byte Hexadecimal Multiply Routine ($DSZM)

ENTRY: From O.$DSBA, O.$DSDB, O.$DS1Z, or O.$DS3L
FUNCTION:
- Multiplies two, 3-byte, hexadecimal numbers. When multiplying, the routine uses a 17-byte work area in this format:



  The multiplicand is placed in the first 7-byte field of the work area, DDSEMA. The multiplier is placed in the 3-byte field called DDSEMB. The product field, DDSEMC, is initially zero.

  The multiplicand is then added to the DDSEMC field a number of times equal to the hexadecimal value in the 4-bit work area (the high-order four bits in DDSEMB). Both DDSEMB and DDSEMC are then shifted left four bits. This places the next four bits of the multiplier in the 4-bit work area and multiplies the value in DDSEMC by 16. This process continues until the low-order four bits of the multiplier are in the 4-bit work area. At this time the multiplicand is added to the value in DDSEMC a number of times equal to the value in the 4-bit work area. This completes the multiplication.
- Moves the product to the output area.

INPUT:
- Address of COMMON in XR1.
- Address of a parameter list in XR2. This list contains:
  1. Address of the 3-byte multiplier.
  2. Address of the 7-byte field where the product is to be placed.
  3. Address of the three bytes containing the multiplicand.

OUTPUT: Final product in the 7-byte field specified by the using phase.
ROUTINES USED: None
EXIT: O.$DSDB, O.$DS1Z, or O.$DS3L

Read Statement from SYSIN Reader Routine (O.$DS9I)

ENTRY: From O.$DSORT, O.$DSBA, O.$DSBC, $DSCA1, or $DSCA2

FUNCTION:

- Places entry point of itself in SYSIN@ located in COMMON.
- Sets bit 1 on in ENVQ2 to indicate the control statements are in the system SYSIN device.
- Checks the return code from the transient SYSIN routine used to read the sort specification statement and does one of the following:
  1. Places the high-order address (leftmost byte) of the statement image in CARD@ located in COMMON if the return code is X'4Ø'.
  2. Gives control to O.$DSGA if the return code is X'6Ø' (indicates an I-O error exists).
  3. Defaults to an end-of-file condition for any other return code. CARD@ points to a '/*ÞÞÞÞÞÞ' image.
  4. Set end-of-file condition if a //END statement is read.

INPUT: Address of COMMON in XR1.
ROUTINES USED: System SYSIN routine
EXIT: O.$DSORT, O.$DSBA, O.$DSBC, $DSCA1, or $DSCA2.

Read Statement from Scheduler Work Area Routine (O.$DS9W)

ENTRY: From O.$DSORT, O.$DSBA, O.$DSBC, or O.$DSCA
FUNCTION:

- Places next entry point in SYSIN@ located in COMMON.
- Sets bit 2 on in ENVQ2 to indicate that the sort specification statements are in the scheduler work area.
- Reads the next specification statement by branching to the Scheduler Work Area (SWA) Get routine. For more information on the SWA Get routine, refer to:
  1. *IBM System/3 Disk Systems System Control Program Logic Manual,* SY21-0502, for Model 6 and Model 10 Disk System
  2. *IBM System/3 Model 12 System Control Program Logic Manual,* for Model 12
  3. *IBM System/3 Model 15 Scheduler Logic Manual,* SY21-0035, for the Model 15
- Checks the return code from the SWA Get routine and does one of the following:
  1. Places the high-order address (leftmost byte) of the card image in CARD@ located in COMMON if the return code is a X'4Ø'.
  2. Places '/*ÞÞÞÞÞÞ' in the first eight bytes of the statement image and sets the address of the statement image in CARD@ if the return code is X'80' (indicates end of file), or if it has read a //END statement.

INPUT: Address of COMMON in XR1.
OUTPUT: None
ROUTINES USED: SWA Get routine
EXIT: O.$DSORT, O.$DSBA, O.$DSBC, O.$DSCA

Read Statement From Source Library Routine (O.$DS9S)

ENTRY: From O.$DSORT, O.$DSBA, O.$DSBC, $DSCA1, or $DSCA2
FUNCTION:

- Places entry point of itself in SYSIN@ located in COMMON.
- Reads the next specification statement by branching to the Source Library Get routine ($$SYSG). For more information on the Source Library Get Routine, refer to:
  1. *IBM System/3 Disk Systems System Control Program Logic Manual,* SY21-0502, for Model 6 or Model 10 Disk System
  2. *IBM System/3 Model 12 System Control Program Logic Manual,* for Model 12
  3. *IBM System/3 Model 15 Scheduler Logic Manual,* SY21-0035, for Model 15
- Checks the function byte from the Source Library Get routine and does one of the following:
  1. If the function byte is a X'02' on the first Get operation, issues a halt ('21' on the Model 10 Disk System, Model 12, and Model 15; 'CD235' on Model 6).
  2. On subsequent Get operations, CARD@ in COMMON is set. CARD@ contains the high-order address (leftmost byte) of the card image.
  3. Places '/*ÞÞÞÞÞÞ' in the first eight bytes of the statement image and sets the address of the statement image in CARD@ if the function byte is a X'01' (indicates end-of-file) or if it has read a // END statement.

INPUT:

- Address of COMMON in XR1.
- Address of a parameter list with the module name and unit (MLIST).

ROUTINES USED: Source Library Get routine ($$SYSG)
EXIT: O.$DSORT, O.$DSBA, O.$DSBC, $DSCA1, or $DSCA2

EXECUTION PHASES

Phase 1L (O.$DS1L)

ENTRY: From O.$DSGA
STORAGE MAP: Figure 3-8
CHART: AA
FUNCTION:

- Performs intialization for PASS#0 common to sort execution modules O.$DS1A, O.$DS1B, and O.$DS1X.
- Loads the required Phase I modules for 1A, 1B, or 1X into storage.
- Determines if input is not from a supported device. If it is not, a terminal error is issued and a halt is received ('2H' halt on Model 10 Disk System, Model 12, and Model 15; 'CD25' halt on Model 6).
- Phase 1L loads O.$DS1B for PASS#0 if there is enough storage for a selection sort (ISTYPE in COMMON is set to S in Phase 0D).
- Loads O.$DS1X for PASS#0 if there is enough storage for a tournament sort.
- Loads O.$DS1A for sort if not enough room in storage for O.$DS1B on O.$DS1X but enough room for a merge exchange sort.
- Initializes the DTF area and DTF name to INPUT (Model 15 only).
- If a card device is used, a switch is set (depending on the record length) for the purpose of padding the input record with blanks during sort execution (Model 12 and Model 15).

INPUT:

- Address of COMMON in XR1.
- Input file.

OUTPUT:

- Initialization that is common to O.$DS1A, O.$DS1B, and O.$DS1X.
- Updated phase load address in COMMON (PHASE@). It is set when Phase 1A, 1B, or 1X gives control to O.$DS1Z.

ROUTINES USED: None
EXIT: Phase 1A, 1B, or 1X

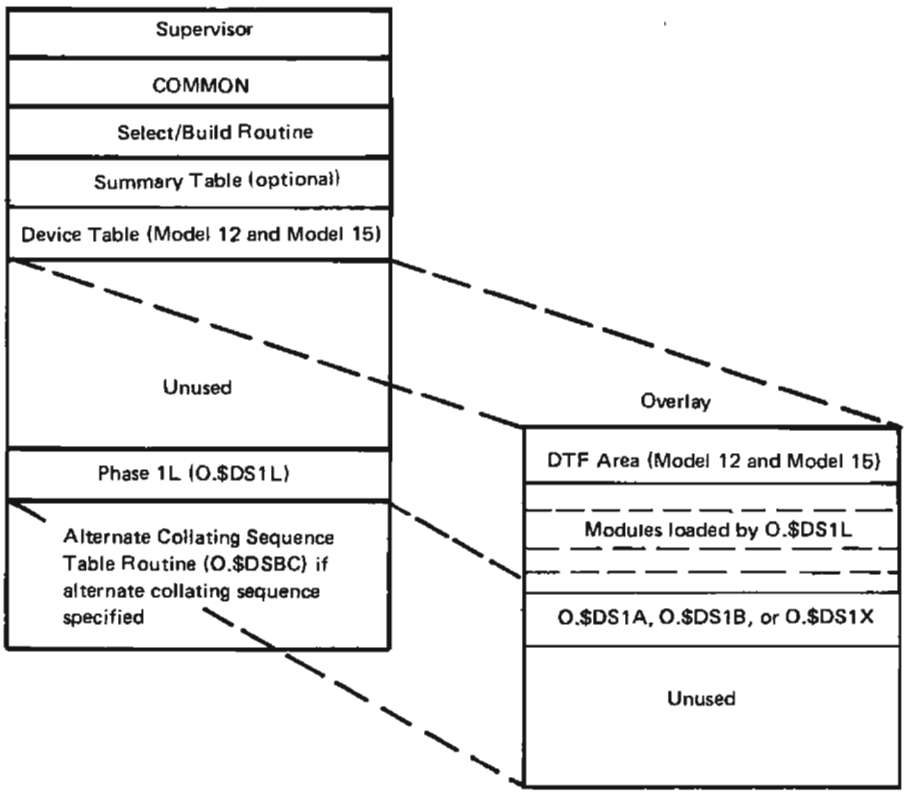


Figure 3-8. Storage Map for Phase 1L

Phase 1A (O.$DS1A)

ENTRY: From O.$DS1L

STORAGE MAPS: Figure 3-9 (Model 6, Model 10 Disk System) Figure 3-10 (Model 12, Model 15)

CHARTS: AB (Model 6, Model 10 Disk System) AC (Model 12 and Model 15).

FUNCTION:

- Sequences records within the work blocks.
- Creates the work file consisting of blocks of sequenced work records (each block constitutes one string).
- Checks for I-O errors; if any errors found, control is passed to O.$DS1Z.
- At end of pass for summary sort, moves the summary table from behind the Select/Build code to immediately behind COMMON.
- Upon entry, passes control to O.$DS1M to determine device type of first file and load its Data Management (Model 12 and Model 15).
- Determines the next device type at the end of each file (up to eight). O.$DS1M is loaded to load Data Management if it is not the same device. (Model 12 and Model 15)
- In the case of same device being a NS or NL tape or a Model 15 card file, calls O.$DS1M to issue halt '26'. If halt response is a skip (1 option), the end of file logic is repeated until either another file is found or the end of the Device Table is reached. (Model 12 and Model 15)

INPUT:

- Address of COMMON in XR1.
- Input file.
- Linkage to modules in O.$DS1L through addresses in COMMON.
- Device Table (Model 12 and Model 15).

OUTPUT: Sequential block (strings) or work records on the work file.

ROUTINES USED: *Model 6 and Model 10 Disk System* – Select/Build Routine, O.$DSBC, O.$DSZA, O.$DS1D, O.$DS8A, O.$DS8B, O.$DS8C, O.$DS8D, O.$DS8F; *Model 12* – Select/Build Routine, O.$DSBC, O.$DSZA, O.$DS1D, O.$DS1M, O.$DS8F, O.$DS9P, O.$$CFUM, O.$$CSIA, O.$$CSIT, O.$$CSUP; *Model 15* – Select/Build Routine, O.$DSBC, O.$DSZA, O.$DS1D, O.$DS1M, O.$DS8F, O.$DS9P, O.$$ARFF, O.$$ARRD, O.$$CFUM, O.$$CPIP, O.$$CSIA, O.$$CSIT, O.$$CSUM, O.$$MFRD, O.$$MMRD

EXIT: O.$DS1Z

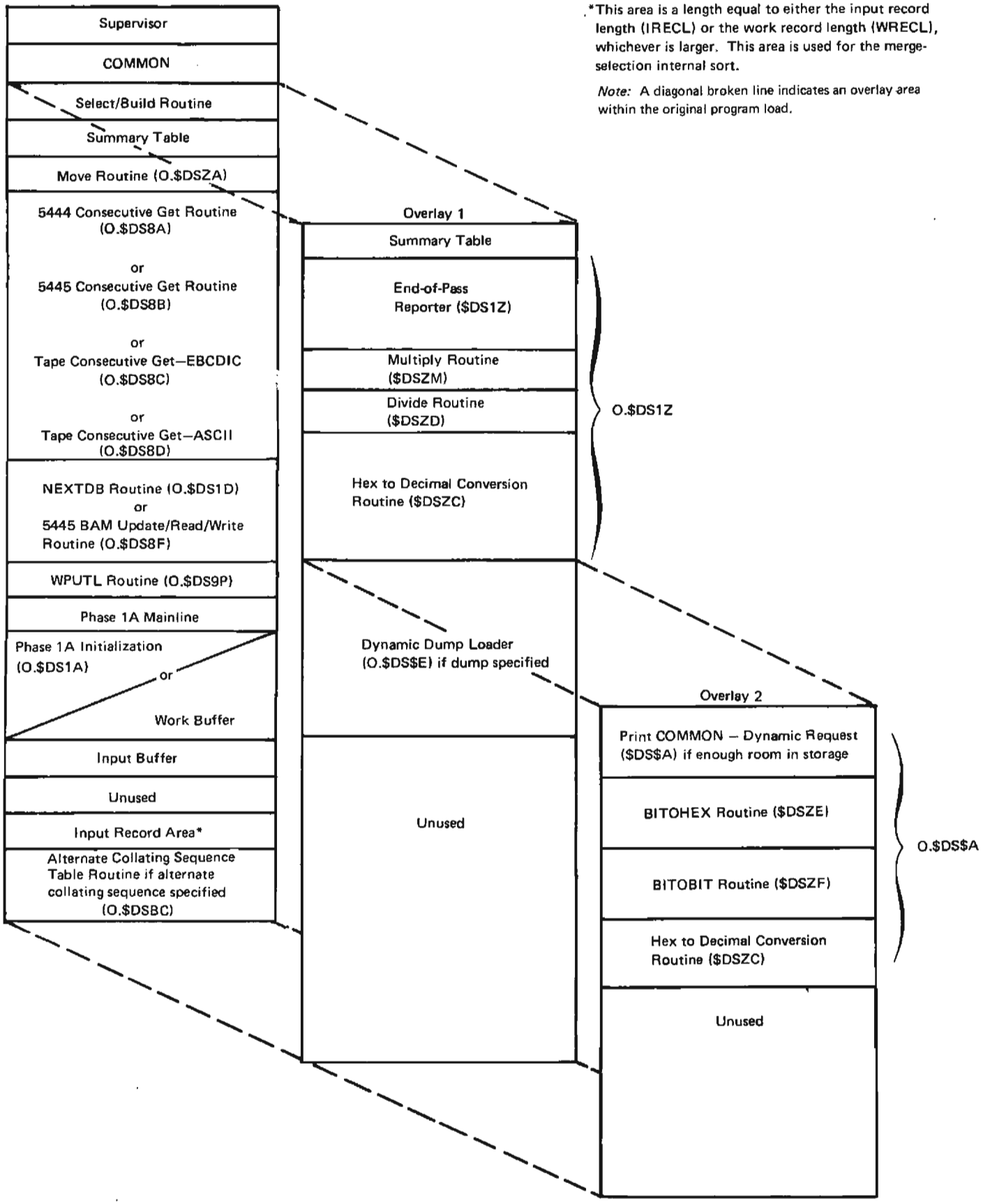


Figure 3-9. Storage Map for Phase 1A – O.$DS1A – (Model 6, Model 10 Disk System)

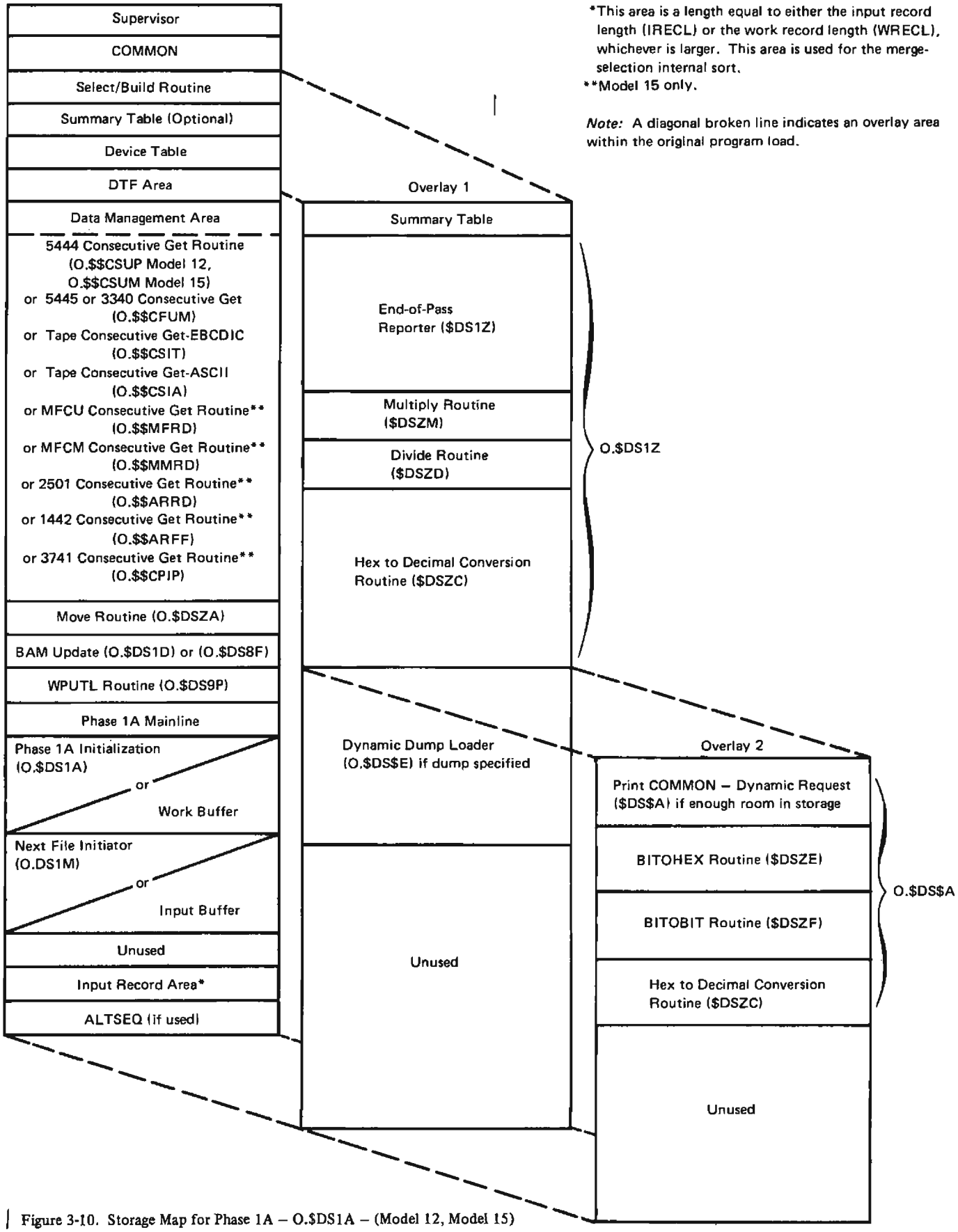


Figure 3-10. Storage Map for Phase 1A – O.$DS1A – (Model 12, Model 15)

Phase 1B (O.$DS1B)

ENTRY: From O.$DS1L
STORAGE MAPS: Figure 3-11 (Model 6, Model 10 Disk System), Figure 3-12 (Model 12, Model 15)
CHARTS: AD (Model 6, Model 10 Disk System) AE (Model 12, Model 15)
FUNCTION:

- Performs a selection internal sort outside of the work block when enough storage is available.
- Produces variable length strings of sequenced work records.
- Places a string of a variable number of work record blocks onto the work file. The work records in each string are in the requested sequence (ascending or descending).
- Checks for a permanent I-O error; if any errors found, control is passed to O.$DS1Z.
- At end of pass for summary sort, moves the summary table from behind the Select/Build code to immediately behind COMMON.
- Upon entry, passes control to O.$DS1M to determine device type of first file and load its Data Management (Model 12 and 15).
- Determines the next device type at the end of each file (up to eight). O.$DS1M is loaded to load Data Management if it is not the same device (Model 12 and Model 15).
- In the case of same device being a NS or NL tape or a Model 15 card file, calls O.$DS1M to issue halt '26'. If halt response is a skip (1 option), the end of file logic is repeated until either another file is found or the end of the Device Table is reached (Model 12 and Model 15).

INPUT:

- Input file.
- Address of COMMON in XR1.
- Linkage to modules in O.$DS1L through addresses in COMMON.
- Device Table (Model 12 and Model 15).

OUTPUT: Variable length strings of work blocks on work file in which the records are in sequence.
ROUTINES USED: *Model 6 and Model 10 Disk System* – Select/Build routine, O.$DSBC, O.$DSZA, O.$DS1D, O.$DS1S, O.$DS8A, O.$DS8B, O.$DS8C, O.$DS8D, O.$DS8F, O.$DS9P; *Model 12* – Select/Build routine, O.$DSBC, O.$DSZA, O.$DS1D, O.$DS1M, O.$DS1S, O.$DS8F, O.$DS9P, O.$$CFUM, O.$$CSIA, O.$$CSIT, O.$$CSUP; *Model 15* – Select/Build routine, O.$DSBC, O.$DSZA, O.$DS1D, O.$DS1M, O.$DS1S, O.$DS8F, O.$DS9P, O.$$ARFF, O.$$ARRD, O.$$CFUM, O.$$CPIP, O.$$CSIA, O.$$CSIT, O.$$CSUM, O.$$MFRD, O.$$MMRD
EXIT: O.$DS1Z

Replacement Selection Internal Sort Routine (O.$DS1S)

ENTRY: From O.$DS1B
STORAGE MAP: Figure 3-11 (Model 6, Model 10 Disk System), Figure 3-12 (Model 12, Model 15)
CHART: EB
FUNCTION:

- Sequences the work records generated by Phase IB.
- Keeps track of the number of records in the internal sort area that are in the current string.
- Produces variable length strings of work records.

INPUT: Work records generated in Phase IB.
OUTPUT: Sequenced work records
ROUTINES USED: None
EXIT: O.$DS1B

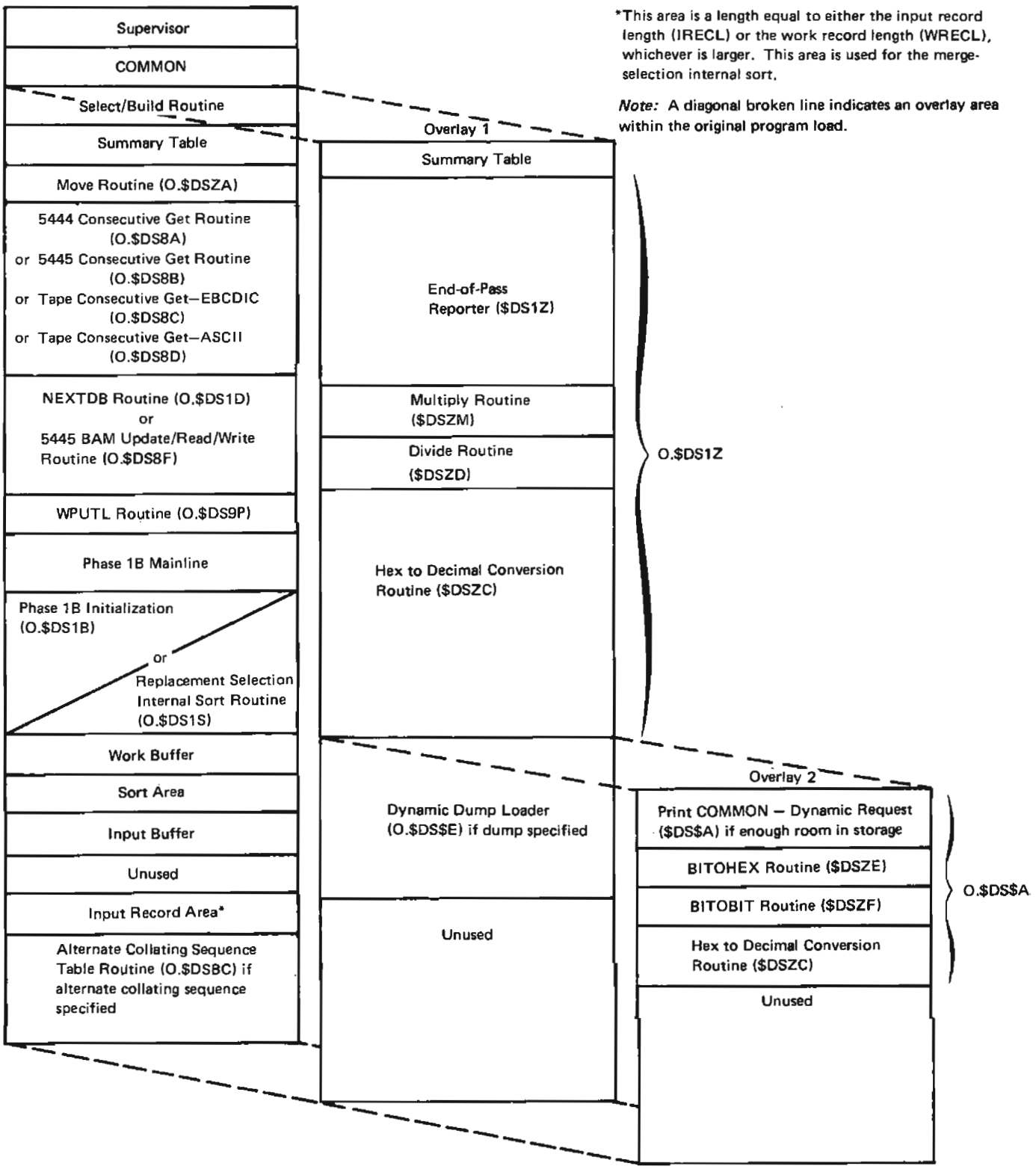
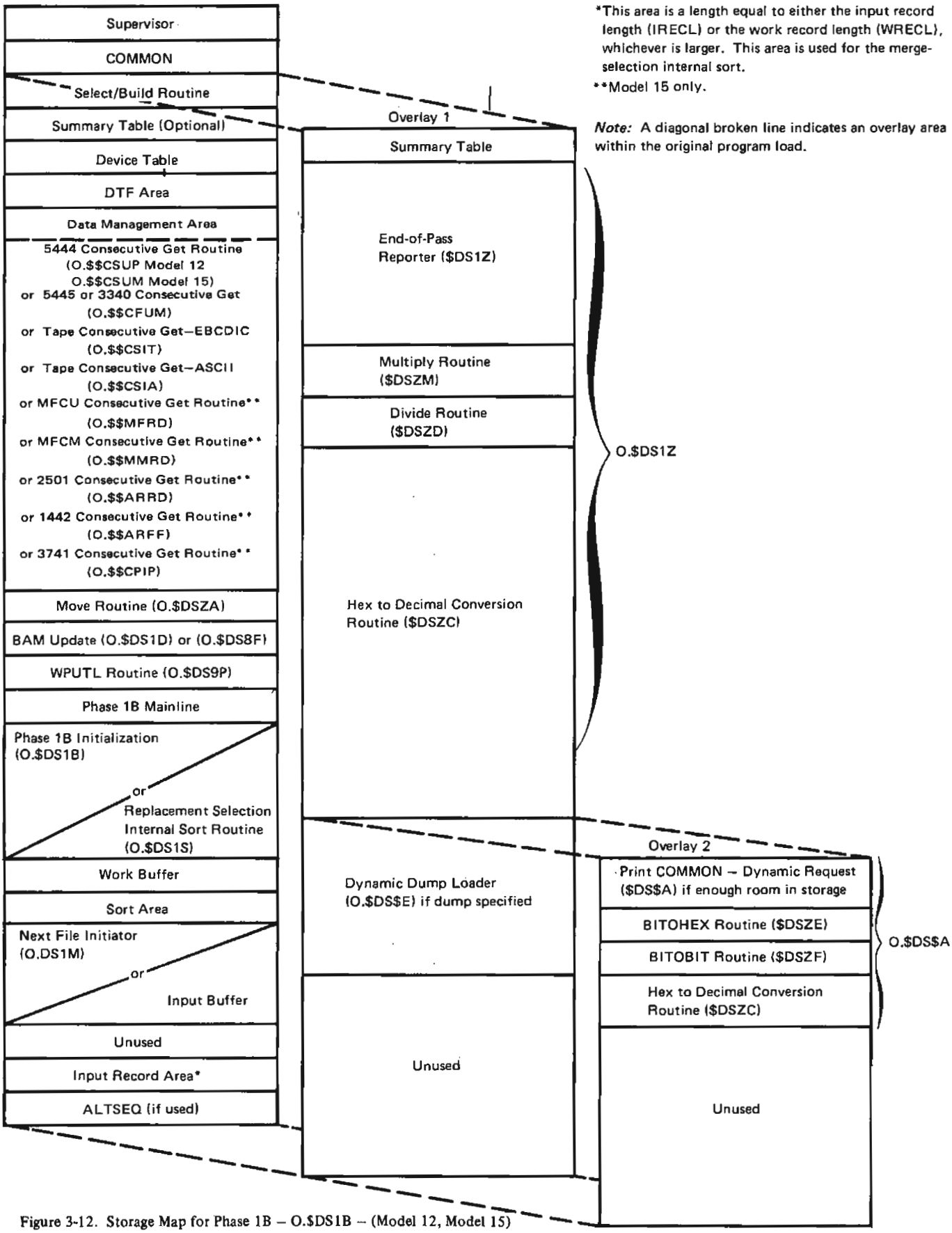


Figure 3-11. Storage Map for Phase 1B O.$DS1B (Model 6, Model 10 Disk System)



*This area is a length equal to either the input record length (IRECL) or the work record length (WRECL), whichever is larger. This area is used for the merge-selection internal sort.

**Model 15 only.

Note: A diagonal broken line indicates an overlay area within the original program load.

Figure 3-12. Storage Map for Phase 1B – O.$DS1B – (Model 12, Model 15)

Phase 1X (O.$DS1X)

ENTRY: From O.$DS1L

STORAGE MAPS: Figure 3-13 (Model 6, Model 10 Disk System), Figure 3-14 (Model 12, Model 15)

CHARTS: AF (Model 6, Model 10 Disk System), AG (Model 12, Model 15)

FUNCTION:

- Performs a tournament sort outside of the work block if it is determined that a tournament sort will run faster than the sort technique used in Phase 1A or 1B.
- Produces variable length strings of sequenced work records.
- Places a string of a variable number of work record blocks onto the work file. The work records in each string are in the requested sequence (ascending or descending).
- Checks for permanent I-O errors; if errors found, control is passed to O.$DS1Z.
- Checks work file size; if too small, control is passed to O.$DS1Z.
- At end of pass for a summary sort, moves the summary table from the next byte after the Select/Build code to the next byte after COMMON.
- Upon entry, passes control to O.$DS1M to determine device type of first file and load its Data Management (Model 12 and Model 15).
- Determines the next device type at the end of each file (up to eight). O.$DS1M is loaded to load Data Management if it is not the same device (Model 12 and Model 15).
- In the case of same device being an NS or NL tape or a Model 15 card file, calls O.$DS1M to issue halt '26'. If halt response is a skip (1 option), the end of file logic is repeated until either another file is found or the end of the Device Table is reached (Model 12 and Model 15).

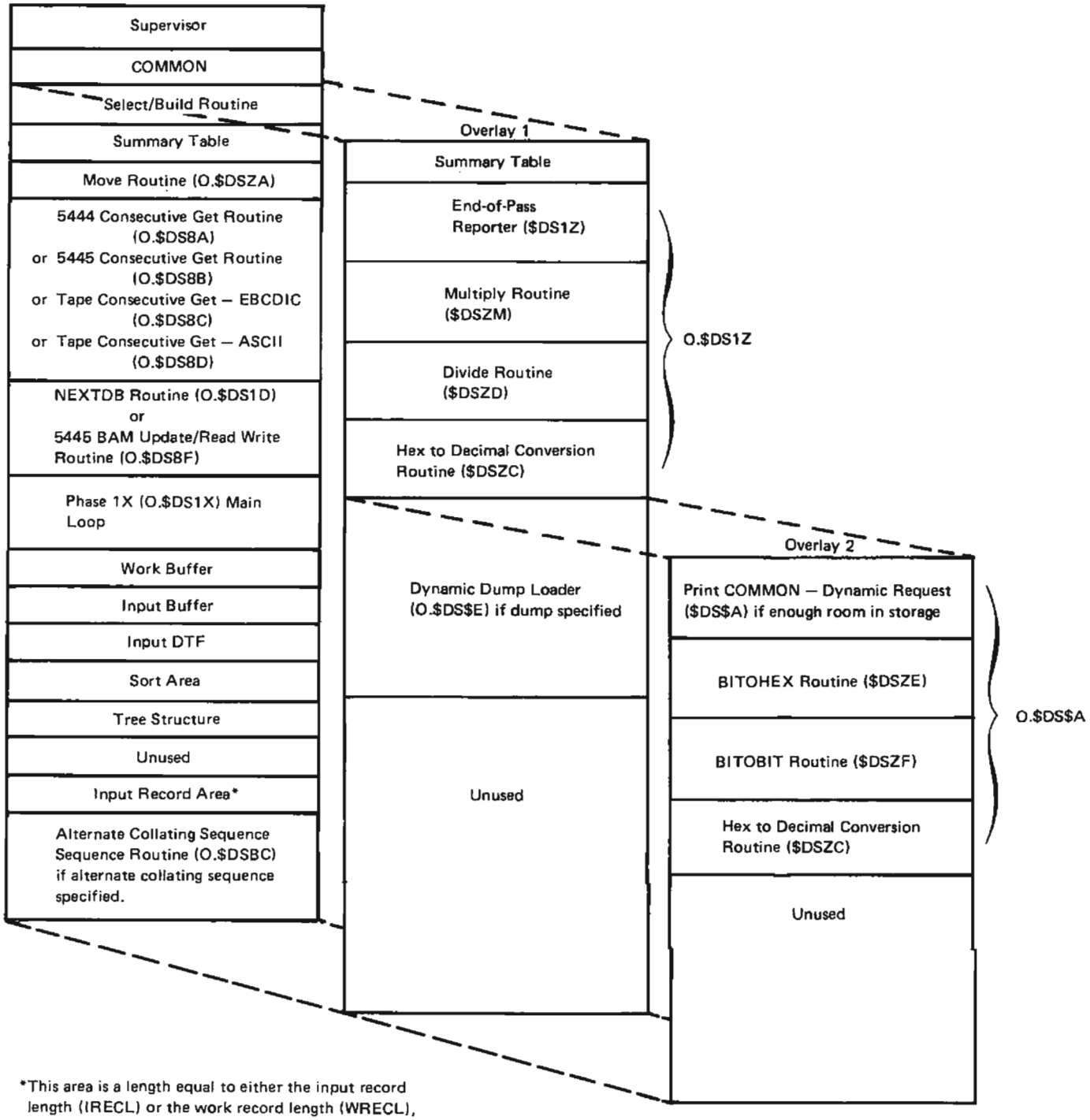
INPUT:

- Input file.
- Address of COMMON in XR1.
- Device Table (Model 12 and Model 15).

OUTPUT: Variable length strings of work blocks on the work file in which the records are in sequence.

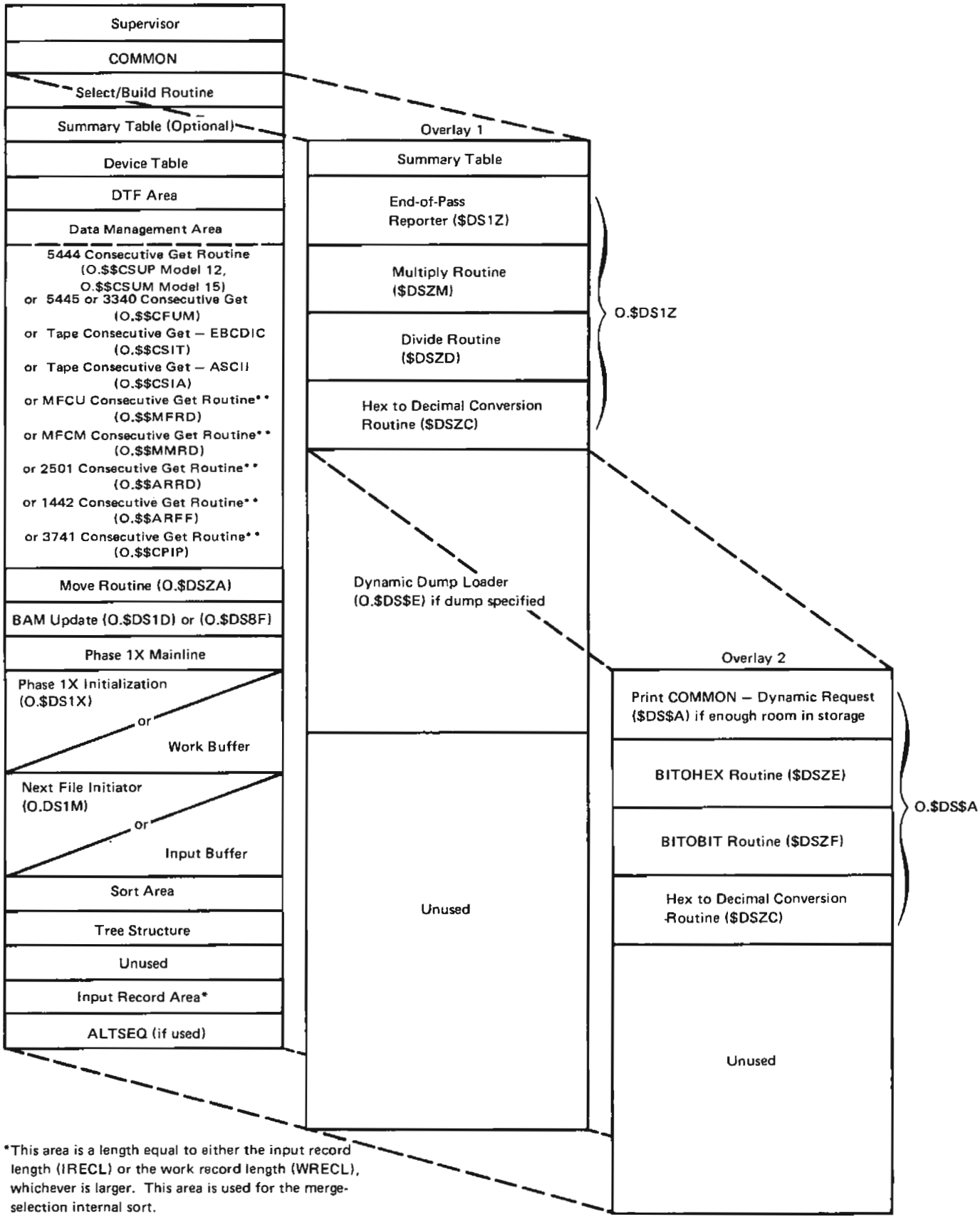
ROUTINE USED: *Model 6 and Model 10 Disk System* – Select/Build routine, O.$DSBC, O.$DSZA, O.$DS1D, O.$DS8A, O.$DS8B, O.$DS8C, O.$DS8D, O.$DS8F; *Model 12* – Select/Build routine, O.$DSBC, O.$DSZA, O.$DS1D, O.$DS1M, O.$DS8F, O.$$CFUM, O.$$CSIA, O.$$CSIT, O.$$CSUP; *Model 15* – Select/Build routine, O.$DSBC, O.$DSZA, O.$DS1D, O.$DS1M, O.$DS8F, O.$$ARFF, O.$$ARRD, O.$$CFUM, O.$$CPIP, O.$$CSIA, O.$$CSIT, O.$$CSUM, O.$$MFRD, O.$$MMRD

EXIT: O.$DS1Z



*This area is a length equal to either the input record length (IRECL) or the work record length (WRECL), whichever is larger. This area is used for the merge-selection internal sort.

Figure 3-13. Storage Map for Phase 1X – O.$DS1X – (Model 6, Model 10 Disk System)



*This area is a length equal to either the input record length (IRECL) or the work record length (WRECL), whichever is larger. This area is used for the merge-selection internal sort.

**Model 15 only.

Figure 3-14. Storage Map for Phase 1X – O.$DS1X – (Model 12, Model 15)

Next File Initiator (O.$DS1M) Model 12 and Model 15

ENTRY: From O.$DS1A, O.$DS1B, or O.$DS1X
CHART: CD (Model 12), CE (Model 15)
FUNCTION:
- Determines the type of device being used and builds the DTF.
- Pads the input record area when the input record length is greater than the record length of the current device (Model 15 only).
- Issues halt '26' for a nonlabeled or nonstandard type or card file.
- Loads Data Management for the current device.

INPUT:
- Device Table.
- Address of COMMON in XR1.

OUTPUT:
- DTF information for opening the file.
- To COMMON: Switch indicating "skip file" when a 1-option is the response to a '26' halt.

ROUTINES USED: O.$DSZA
EXIT: O.$DS1A, O.$DS1B, or O.$DS1X

End-of-Pass Reporter (O.$DS1Z)

ENTRY: From O.$DS1A, O.$DS1B, O.$DS1X, O.$DS2A, O.$DS3A, or O.$DS3S
STORAGE MAP: See the storage maps of the appropriate phase.
CHARTS: DA (Model 6, Model 10 Disk System) DB (Model 12, Model 15)
FUNCTION:

- Performs the following for pass number ∅ (Phase I end):
  1. Calculates the number of tracks the work file uses.
  2. Calculates the number of passes remaining.
  3. Sets WREC#T (total number of work records created) in COMMON.
  4. Sets WBLK#T (total number of work blocks created) in COMMON.
  5. Logs the number of work records read and accepted if the print entry option is not 3.
  6. Determines if there were any records in the input file; if none, control is passed to O.$DS4A.
  7. Determines if no records were selected.
  8. Determines if there are more work records than input records.
  9. Allocates space for the output file if deferred after the input file is closed.
  10. Calculates output file size (in tracks).
  11. Compares the actual output file size with the output file size required; gives a halt if the actual output file is too small.
  12. Selects a merge configuration based on the number of strings created (Model 12 and Model 15).
- Performs the following for last pass (Phase III end):
  1. Determines if the number of work records is equal to the total number of work records. If they are not equal, this is a terminal error and control is passed to O.$DS4A.
  2. Writes a message giving the number of sorted records on the output file if the print option entry is ∅ or 1.
- Performs the following for all other passes:
  1. Prints end-of-pass messages if the print option entry is ∅ or 1.
  2. Ensures that the current record counter (WREC# in COMMON) is not higher than the total number of work records created (WREC#T in COMMON).
  3. Determines whether to perform an intermediate pass (Phase II) or a last pass (Phase III).

INPUT: Address of COMMON in XR1.
OUTPUT:

- Error messages.
- Information messages.

ROUTINES USED: $DSZD, $DSZM, $DSZC
EXIT: O.$DS2L, O.$DS2A (Model 12 and Model 15), O.$DS3L, or O.$DS4A

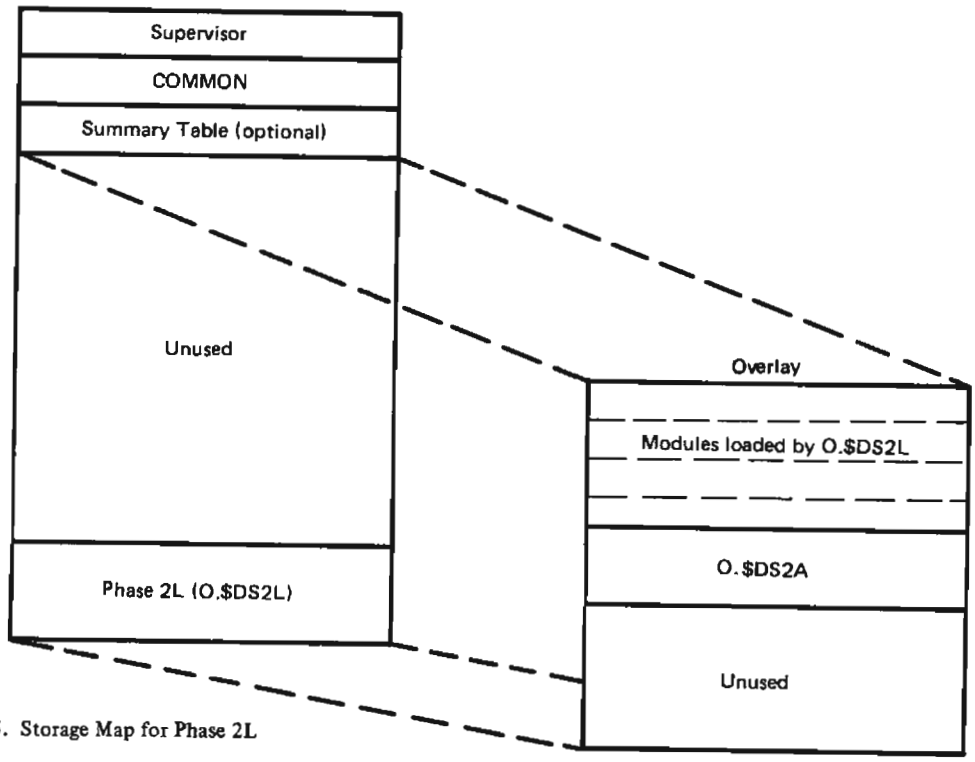


Figure 3-15. Storage Map for Phase 2L

Phase 2L (O.$DS2L)

ENTRY: From O.$DS1Z
STORAGE MAP: Figure 3-15
FUNCTION:
- Loads the modules required by O.$DS2A to perform Phase II into storage.
- Ensures that the work file is using 5444 or 5445; if it is not, control is passed to the EOJ Transient.
- Exchanges control to O.$DS2A when required modules are loaded.

INPUT: Address of COMMON in XR1.
OUTPUT: Modules required by O.$DS2A are in storage and their entry points are set up in COMMON.
ROUTINES USED: None
EXIT: O.$DS2A

Phase 2A (O.$DS2A)

ENTRY: From O.$DS2L and O.$DS1Z
STORAGE MAPS: Figure 16 (Model 6, Model 10 Disk System), Figure 17 (Model 12 and Model 15)
CHART: BA
FUNCTION:
- Sets order of merge for this pass (OMP) to the normal order of merge (OM).
- Determines work block addresses (WBLK@).
- Determines output block addresses (WBUF@).
- Calls O.$DS9G to locate the next record to be merged.
- Calls O.$DS9P to locate the position of the next record in the work block.
- Places the current CSD/CHR into the WCBLKC field in COMMON.
- Places the next CSD/CHR into WNBLKC in COMMON.
- Merges strings created in Phase IA or IB until the total number of strings is less than or equal to the sort's order of merge.

INPUT:
- Work file.
- OLDS structured element array (see *Section 5. Data Area Formats* for a description of this table).
- AVAIL table (see *Section 5. Data Area Formats* for a description of this table).

OUTPUT: Strings of work records less than or equal to the order of merge on the work file.
ROUTINES USED: O.$DS8E, O.$DS9G, O.$DS9P
EXIT: O.$DS1Z

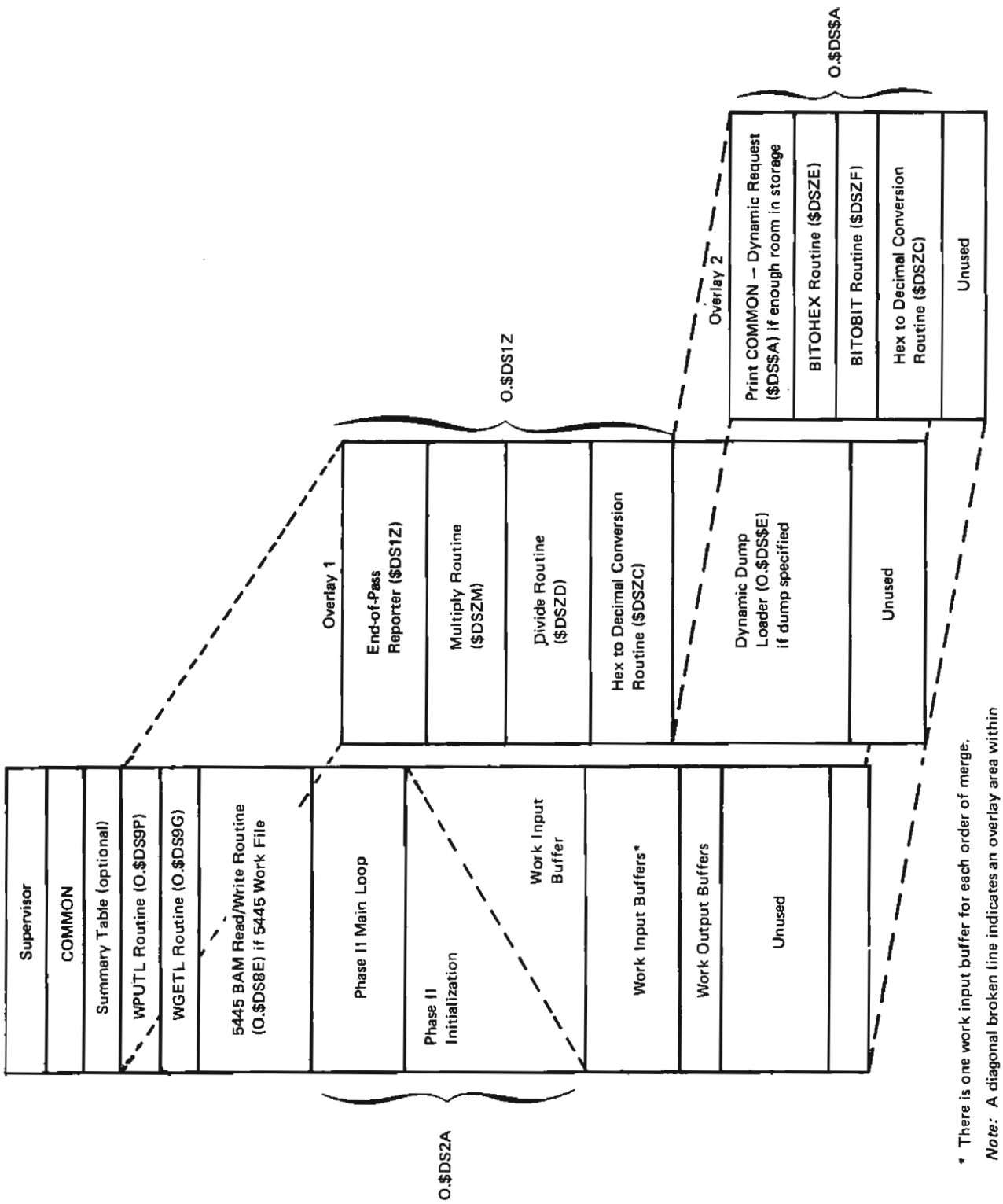


Figure 3-16. Storage Map for Phase 2A – O.$DS2A – (Model 6, Model 10 Disk System)

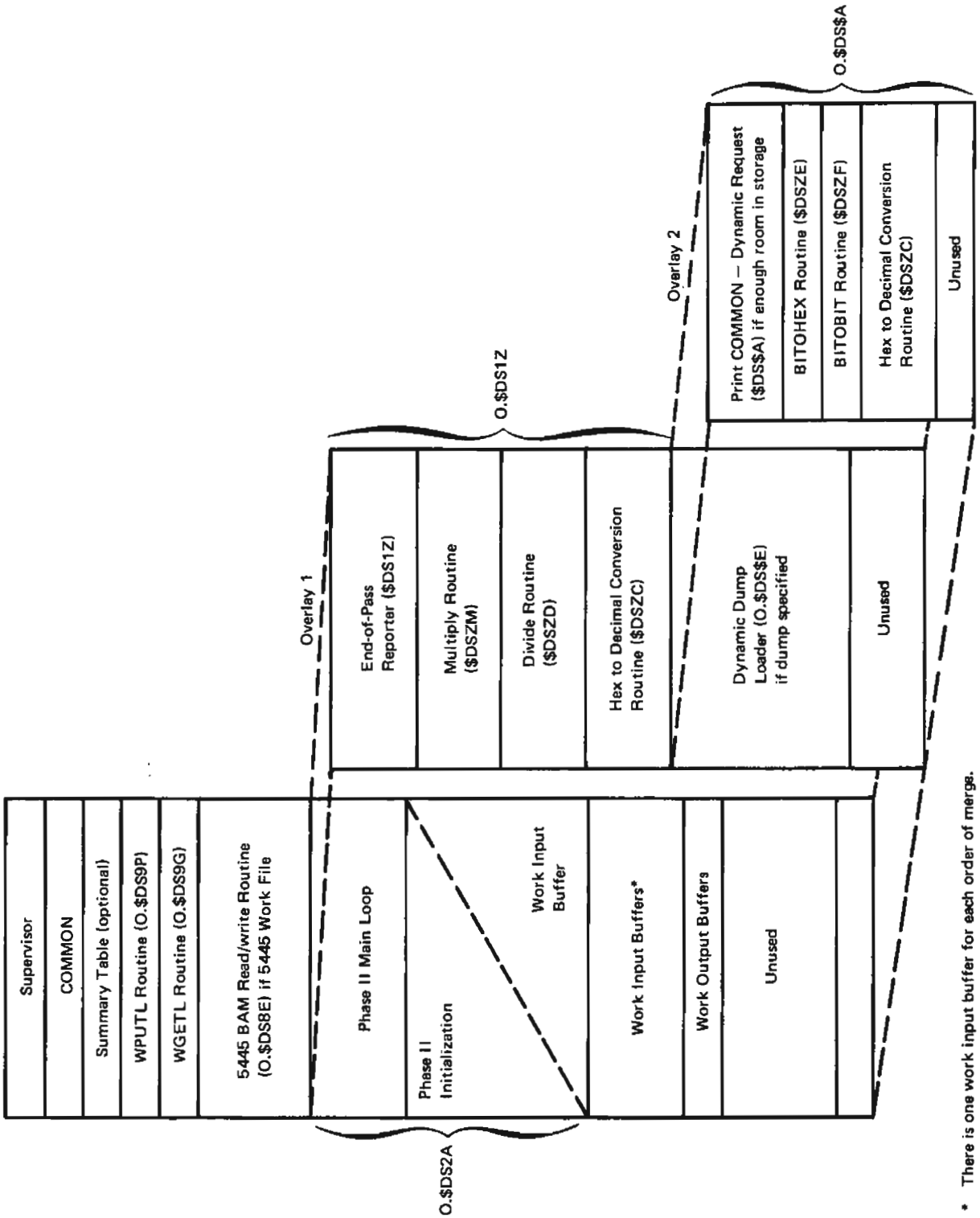


Figure 3-17. Storage Map for Phase 2A – O.$DS2A – (Model 12, Model 15)

Phase 3L (O.$DS3L)

ENTRY: From O.$DS1Z
STORAGE MAP: Figure 3-18
CHART: CA
FUNCTION:

- Loads the modules required by O.$DS3A/O.$DS3S to perform Phase III into storage.
- Recomputes the output buffer number and length if the number of remaining strings is less than the order of merge.
- Determines if there is enough storage available for last pass; if not, this is an internal sort error and control is passed to EOJ Transient.
- Performs initialization for the last pass common to both summary and non-summary Phase III modules, O.$DS3A and O.$DS3S.
- Loads O.$DS3A into storage if the sort is non-summary. If it is a summary sort O.$DS3S is loaded into storage.

INPUT: Address of COMMON in XR1.
OUTPUT: Initialization common to O.$DS3A and O.$DS3S, including loading of required Phase III modules.
ROUTINES USED: $DSZD, $DSZM (O.$DS3L loads in the modules needed for O.$DS3A and O.$DS3S to conserve space on the system disk).
EXIT: O.$DS3A or O.$DS3S

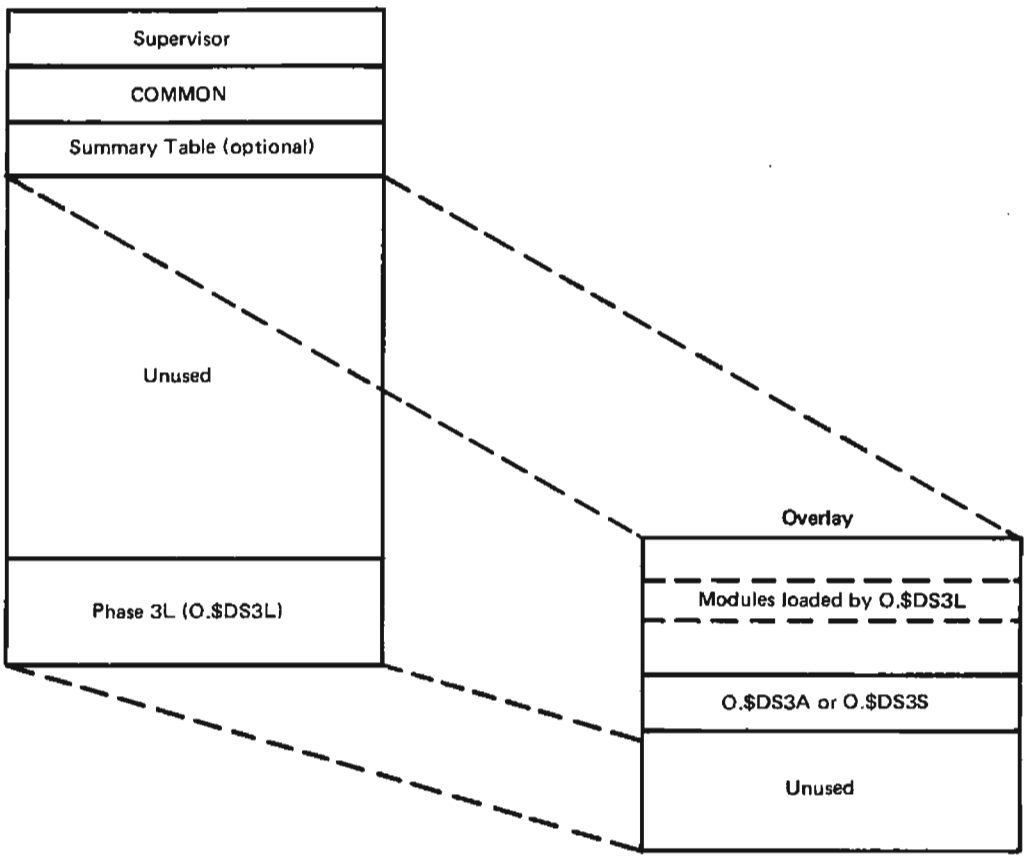


Figure 3-18. Storage Map for Phase 3L

Phase 3A (O.$DS3A)

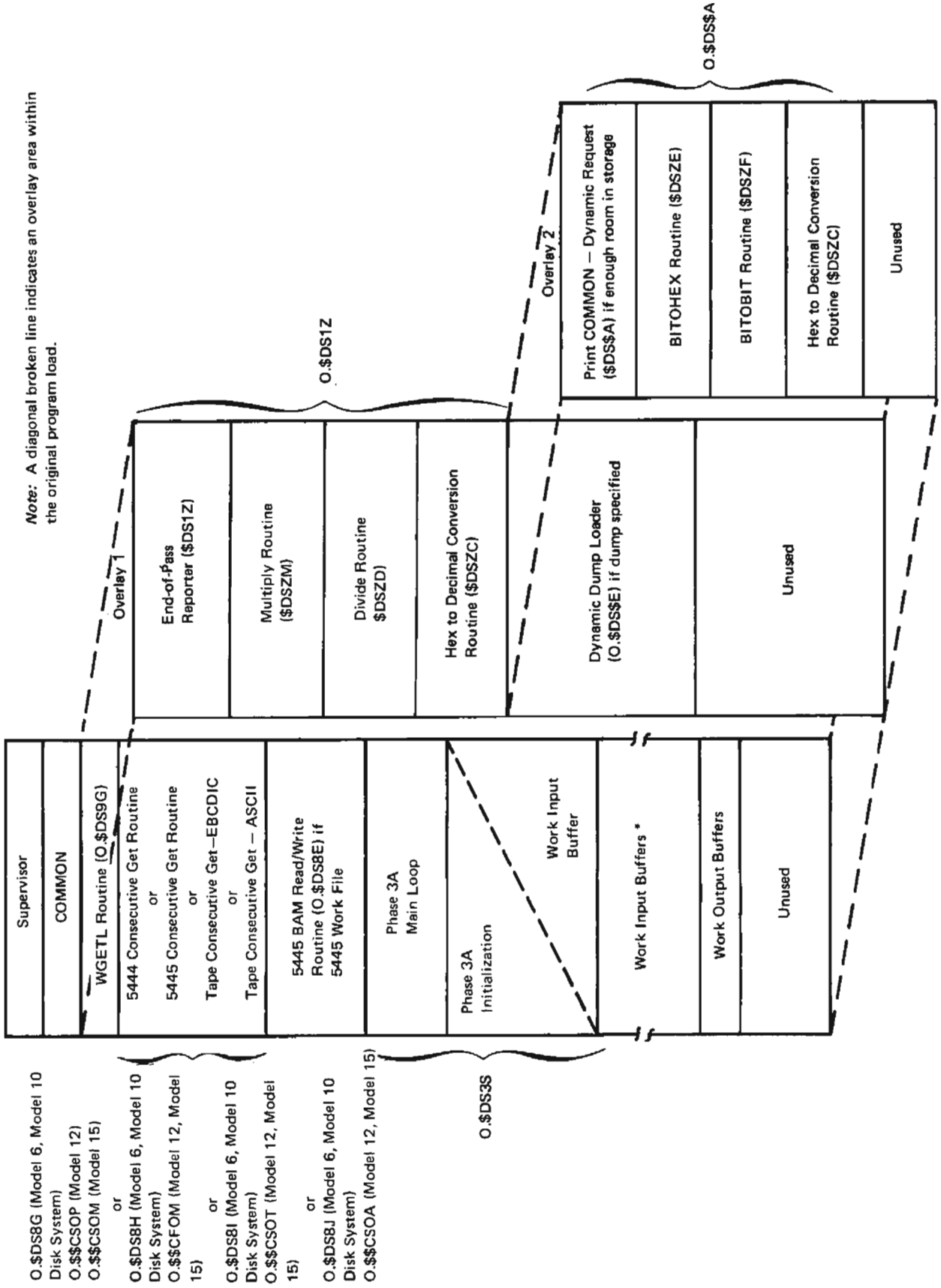
ENTRY: From O.$DS3L
STORAGE MAP: Figure 3-19
CHART: CB
FUNCTION:

- Performs last pass of the merge started in Phase I or Phase II, depending on where the number of remaining strings became less than the order of merge.
- If this is a SORTA with tape output, blocks six ADDROUT records to each 18-byte tape record.
- Places the sorted records consecutively in the output file.
- Determines if end of extent was reached in Phase III; if it was, control is passed to O.$DS1Z.
- Checks for a permanent I-O error during a Put request to the output file; if error found, control is passed to O.$DS1Z.
- Determines if error occurred in closing the output file; if it did, control is passed to O.$DS1Z.

INPUT:

- Work file.
- Address of COMMON in XR1.
- Linkage to modules loaded by O.$DS3L through addresses in COMMON.

OUTPUT: Sorted output file.
ROUTINES USED: *Model 6 and Model 10 Disk System* – O.$DS8E, O.$DS8G, O.$DS8H, O.DS8I, O.$DS8J, O.$DS9G; *Model 12* – O.$DS8E, O.$DS9G, O.$$CFOM, O.$$CSOA, O.$$CSOP, O.$$CSOT; *Model 15* – O.$DS8E, O.$$CSOM, O.$$CFOM, O.$$CSOT, O.$$CSOA, O.$DS9G
EXIT: O.$DS1Z



* There is one work input buffer for each order of merge.

Note: A diagonal broken line indicates an overlay area within the original program load.

● Figure 3-19. Storage Map for Phase 3A – O.$DS3A – (All Models)

Phase 3S (O.$DS3S)

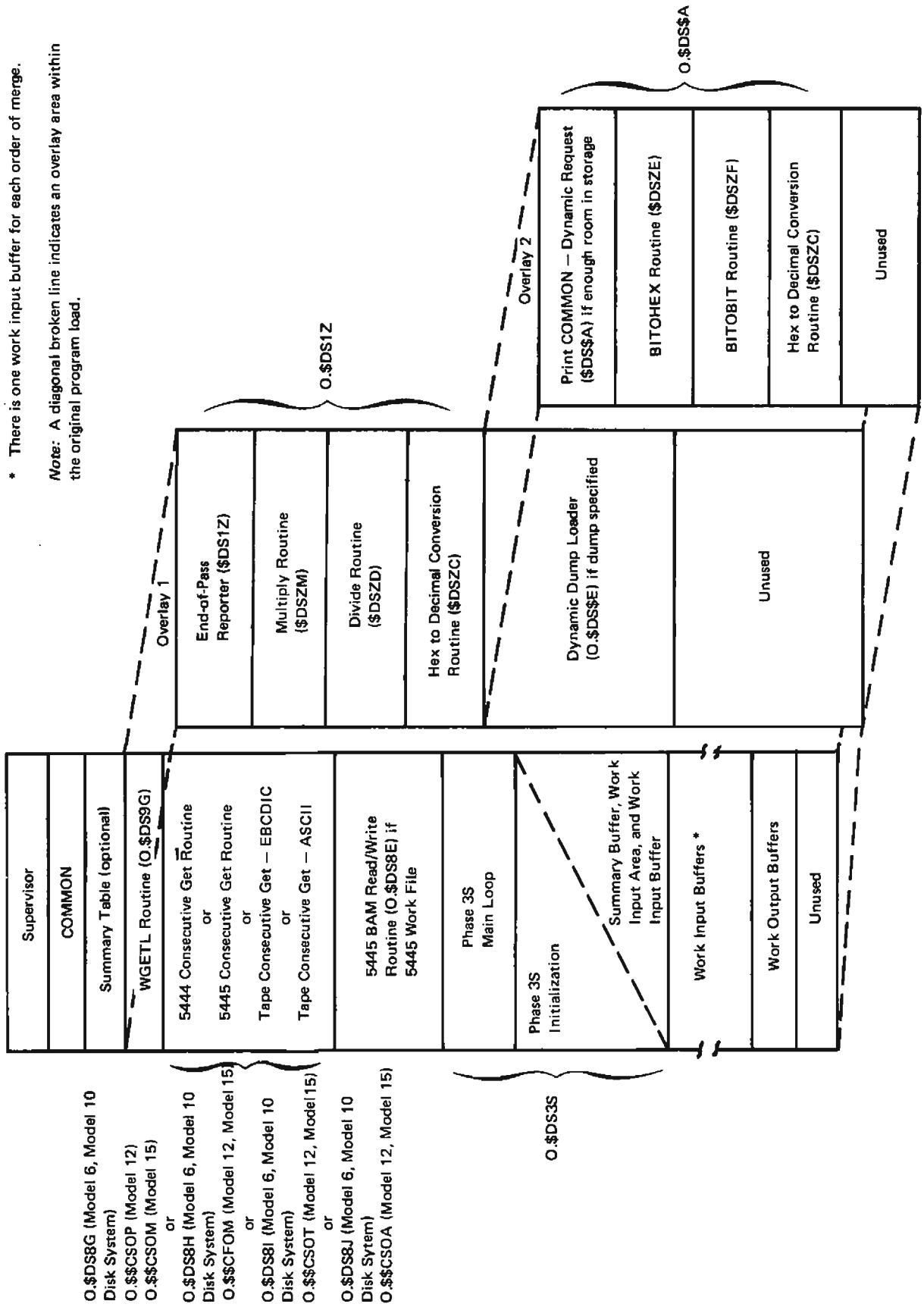
ENTRY: From O.$DS3L
STORAGE MAP: Figure 3-20
CHART: CC
FUNCTION:

- Performs last pass of the merge started in Phase I or Phase II, depending on where the number of remaining strings became less than the order of merge.
- Deletes duplicate records if no summary table exists.
- Summarizes duplicate records as specified if a summary table exists.
- Determines if records are in proper sequence; if not, control is passed to the EOJ Transient.
- Checks for end of extent; if reached, control is passed to O.$DS1Z.
- Determines if an error occurred while closing the output file; if it did, control is passed to O.$DS1Z.
- Places the sorted records consecutively in the output file.

INPUT:

- Work file.
- Address of COMMON in XR1.
- Summary table.
- Linkage to modules loaded by O.$DS3L through addresses in COMMON.

OUTPUT: Addresses in COMMON.
ROUTINES USED: *Model 6 and Model 10 Disk System* – O.$DS8E, O.$DS8G, O.$DS8H, O.$DS8I, O.$DS8J, O.$DS9G; *Model 12* – O.$DS8E, O.$DS9G, O.$$CFOM, O.$$CSOA, O.$$CSOP, O.$$CSOT; *Model 15* – O.$DS8E, O.$$CSOM, O.$$CFOM, O.$$CSOT, O.$$CSOA, O.$DS9G
EXIT: O.$DS1Z



● Figure 3-20. Storage Map for Phase 3S – O.$DS3S – (All Models)

Phase 4 (O.$DS4A)

ENTRY: From O.$DS1Z
STORAGE MAP: Figure 3-21
FUNCTION:
- Prints I-O error or terminal error action messages if the print option entry is not 3.
- Displays the halt and ends the job after unsuccessful completion via the Halt/Syslog Transient routine.
- Terminates the Disk Sort program after successful completion via Halt/Syslog Transient routine

INPUT: Address of COMMON in XR1.
OUTPUT:
- Printed message indicating whether end of job is normal or abnormal.
- Printed action messages if the print option entry is not 3 and errors were found.
- Program halt display for errors found.

ROUTINES USED: None
EXIT: EOJ Transient

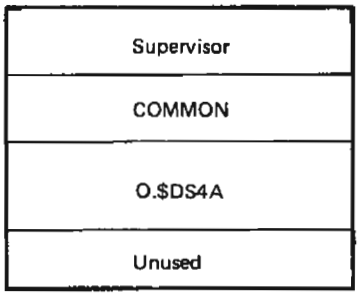


Figure 3-21. Storage Map for Phase 4

EXECUTION ROUTINES

Variable Length Move Routine (O.$DSZA)

ENTRY: From O.$DS1A, O.$DS1B, or O.$DS1X
FUNCTION: Moves a given number of bytes from one area in storage to another.
INPUT:
- Address of COMMON in XR1.
- Address of a parameter list in XR2. This list contains: the high-order address of the area in storage where the bytes are to be moved; the high-order address of the area from which the bytes are to be moved; the address of a two-byte field that gives the number of bytes to be moved.

OUTPUT: The moved bytes to the storage address indicated in the parameter list.
ROUTINES USED: None
EXIT: O.$DS1A, O.$DS1B, or O.$DS1X

NEXTDB Routine for Phase 1 (O.$DS1D)

ENTRY: From O.$DS9P or O.$DS1X
CHART: EA
FUNCTION:
- Calculates and places in COMMON the cylinder/sector/displacement (CSD) of the current work record.
- Calculates and places in COMMON the CSD of the next work record.
- Starts the next CSD on the next extent if block length plus displacement is greater than block length in the table of work file extents. This is done because a partial block is not used.
- Checks whether the next block used is in an existing extent; if not, control is passed to the next sequential instruction + 1 of O.$DS9P or O.$DS1X.

INPUT:
- Table of work file extents.
- Block length.
- Address of COMMON in XR1.

OUTPUT: Entries in COMMON.
ROUTINES USED: None
EXIT: O.$DS9P or O.$DS1X

Work File Put-Locate (WPUTL) Routine (O.$DS9P)

ENTRY: From O.$DS1A, O.$DS2A, or O.$DS1B
CHARTS: EE (Model 6, Model 10 Disk System) EF (Model 12, Model 15)
FUNCTION: Figure 3-22 is a chart showing, in general terms, how WPUTL is used by the execution phases. This routine:

- Maintains interblock vectors within a string of blocks.
- Maintains interstring vectors in the work file.
- Analyzes input and performs one of these functions:
  1. Sets up WBLK@, WRITE@, and WRITE.
  2. Writes the first record in the first block of the first string and indicates if it is the only record in the block.
  3. Checks for error during the WRITE; if error found, control is passed to O.$DS1A, O.$DS2A, or O.$DS1B.
  4. Writes the first record in the first block of the first string and indicates if it is the only record in the block.
  5. Writes the next record in the current new block and indicates if it is the last record in the block.
  6. Writes out a full block continuing the current new string.
  7. Writes out a full block ending the current new string.
  8. Writes out a full block and ends processing of this current new string (Phase 2A only).
  9. Writes out any records in the current new block and indicates end of pass.
  10. Checks if work file is too small; if it is, control is passed to O.$DS1A, O.$DS1B, or O.$DS2A.

INPUT: Address of COMMON in XR1.
OUTPUT: Figure 3-23 shows the possible combinations of input and output fields and their meaning on entry and exit from WPUTL.
ROUTINES USED: None
EXIT: O.$DS1A, O.$DS1B, or O.$DS2A

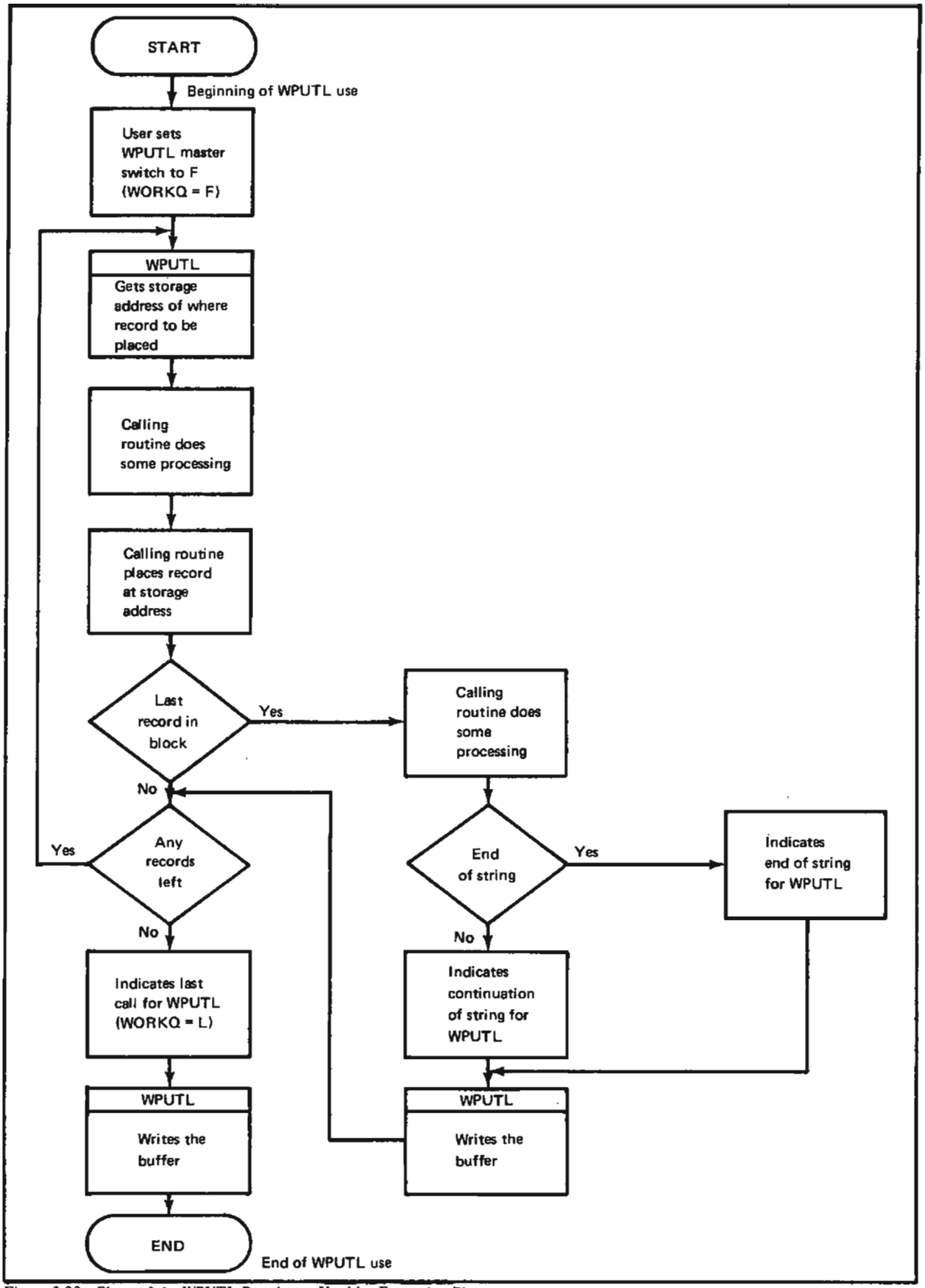


Figure 3-22. Chart of the WPUTL Routine as Used by Execution Phases

Function		Input/Output I-Input O-Output	WORKQ F-First request I-Intermediate request L-All done	WRECQ F-Next record is first in block I-Last record is not last in block L-Last record is last in block	WSTRQ C-Continue current string E-End of current string	WBLKQ N-Start new string O-Continue
First Call		I O	F I	- I or L	- C	- N
Put-locate	First record in first block of new string	I O	I I	F I or L	C C	N N
	First record in intermediate block of continuing new string	I O	I I	F I or L	C C	O O
	Not first record in first block of new string	I O	I I	I I or L	C C	N N
	Not first record in intermediate block of continuing new string	I O	I I	I I or L	C C	O O
Write: continue string	First block of string	I O	I I	L F	C C	N O
	Intermediate block of string	I O	I I	L F	C C	O O
Write: end the string	First and only	I O	I I	L F	E C	N N
	Other block before in string	I O	I I	L F	E C	O N
Write: discontinue (Phase II only)	(NEXTDB resets WSTRQ)	I O	L L	L -	E C	N or O N
Close	Just ended a string	I	L	F	C	N
	At block boundary, string continues	I	L	F	C	O
	Within first block of string	I	L	I	C	N
	Within intermediate block of string	I	L	I	C	O

Figure 3-23. Combinations of Input and Output Fields

Work File Get-Locate (WGETL) Routine (O.$DS9G)

ENTRY: From O.$DS2A, O.$DS3A, or O.$DS3S
CHART: ED
FUNCTION:

- Provides the logical get I-O function for the work file.
- Determines if INQ=F (first call of pass). If it does, the CSD/CHR of the first string in WGETL is the CSD/CHR of the last string completed by WPUTL in the previous pass. A branch is then taken to read the block.
- Checks for errors during read; if error found, control is passed to O.$DS2A, O.$DS3A, or O.$DS3S with the error marked.
- Determines if INQ=F (first call of this pass). If it does, sets up next string and availability table, then control is passed to $DS9R.
- Determines if INQ=N (starting a new string); if it does, checks if old strings are left. If there are none left, the order of merge of new string is Ø and control is passed to O.$DS2A, O.$DS3A, or O.$DS3S. If old strings are left, another block is read.
- Checks if all records in block are processed. If not, finds next record address; then control is passed to O.$DS2A, O.$DS3A, or O.$DS3S.
- Reads new block and sets pointers to next block in old string.

INPUT: Address of COMMON in XR1.
OUTPUT:

- OLDS structured element array.
- AVAIL table.

ROUTINES USED: O.$DS9G, $DS9R
EXIT: O.$DS2A, O.$DS3A, or O.$DS3S

DISK SORT DATA MANAGEMENT ROUTINES

The Disk Sort program uses several data management functions. Each of these functions is part of a load module containing the 6-byte module ID, the necessary data management code for the functions, a maintenance area, and the entry point of the load module (see Figure 3-24).

Figure 3-25 lists the Disk Sort Data Management Modules and the entry point, functions, input and output of each. In addition, the phases and routines which call each Data Management routine are listed.

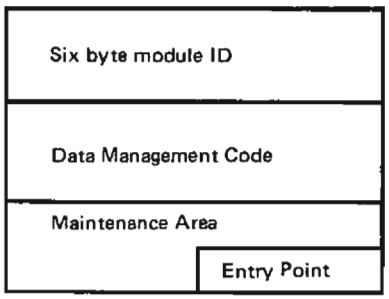


Figure 3-24. Disk Sort Data Management Load Module Organization

Disk Sort Data Management Routine	Entry Point		Data Management Routine *	Called By:
	Model 6, Model 10 Disk System	Model 12, Model 15		
5444 Data Management Consecutive Get routine (Locate Mode) — O.$DS8A	$DS7A0	$$CSUP (Model 12 only) $$CSUM (Model 15 only)	$$CSUP (Model 12 only) $$CSUM** (Model 15 only)	O.$DS1A O.$DS1B O.$DS1X
5445 Data Management Consecutive Get routine (Locate Mode) — O.$DS8B	$DS7B0	$$CFUM****	$$CFUM**	O.$DS1A O.$DS1B O.$DS1X
5445 Data Management BAM Read/Write routine — O.$DS8E	$DS7E0	$DS7G0****	$$CFRW	O.$DS9G O.$DS9P
5445 Data Management BAM Update/Read/ Write routine — O.$DS8F	$DS7F0 (Read/Write) $DS7F0-6 (Update)	$DS7F0 (Read/Write) $DS7F0-6 (Update)****	$$CFDT	O.$DS9P O.$DS1X
5444 Data Management Consecutive Put routine — O.$DS8G	$DS7G0	$$CSOM (Model 15 only) $$CSOP (Model 12 only)	$$CSOM (Model 15 only) $$CSOP (Model 12 only)	O.$DS3A O.$DS3S
5445 Data Management Consecutive Put routine — O.$DS8H	$DS7H0	$$CFOM****	$$CFOM	O.$DS3A O.$DS3S
Tape Data Management Consecutive Get EBCDIC routine (Locate Mode) — O.$DS8C	$DS8C0	$$CSIT	$$CSIT***	O.$DS1A O.$DS1B O.$DS1X
Tape Data Management Consecutive Get ASCII routine (Locate Mode) — O.$DS8D	$DS8D0	$$CSIA	$$CSIA	O.$DS1A O.$DS1B O.$DS1X
Tape Data Management Consecutive Put EBCDIC Routine — O.$DS8I	$DS8I	$$CSOT	$$CSOT***	O.$DS3A O.$DS3S
Tape Data Management Consecutive Put ASCII Routine — O.$DS8J	$DS8J	$$CSOA	$$CSOA	O.$DS3A O.$DS3S

Figure 3-25 (Part 1 of 2). Disk Sort Data Management Routines

Disk Sort Data Management Routine	Entry Point		Data Management Routine *	Called By:
	Model 6, Model 10 Disk System, Model 12	Model 15		
MFCU Read – O.$$MFRD	Not supported	$$MFRD	$$MFRD	O.$DS1A O.$DS1B O.$DS1X
MFCM Read – O.$$MMRD	Not supported	$$MMRD	$$MMRD	O.$DS1A O.$DS1B O.$DS1X
2501 Read – O.$$ARRD	Not supported	$$ARRD	$$ARRD	O.$DS1A O.$DS1B O.$DS1X
1442 Read/Punch – O.$$ARFF	Not supported	$$ARFF	$$ARFF	O.$DS1A O.$DS1B O.$DS1X
3741 Data Station –	Not supported	$$CPIP	$$CPIP	O.$DS1A O.$DS1B O.$DS1X

*Corresponding Disk Sort Data Management Routine performs same functions, requires same input, and produces same output as Data Management routine in this column. For Model 6 and Model 10 Disk System, see *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual,* SY21-0512; for Model 12, see *IBM System/3 Model 12 System Control Program Logic Manual;* for Model 15, see *IBM System/3 Model 15 Data Management Logic Manual,* SY21-0034. If any data management errors are found for these routines, control returns to the calling routine.

**Input only

***This Data Management routine is used for both 9-track EBCDIC tapes and 7-track tapes.

****Also used for 3340 Data Management

Figure 3-25 (Part 2 of 2). Disk Sort Data Management Routines

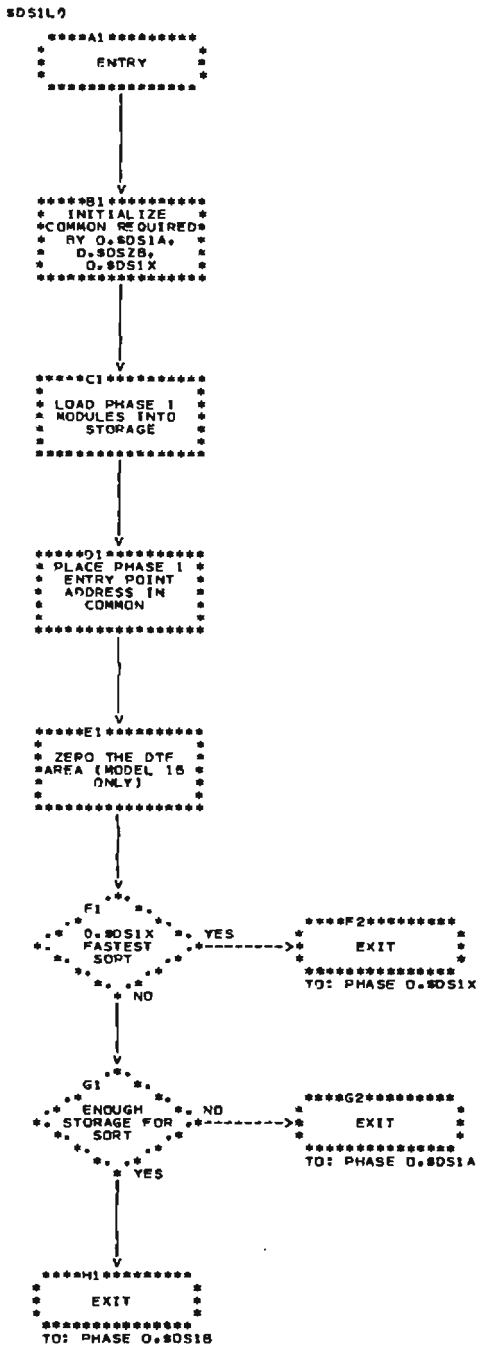


Chart AA. Phase 1L

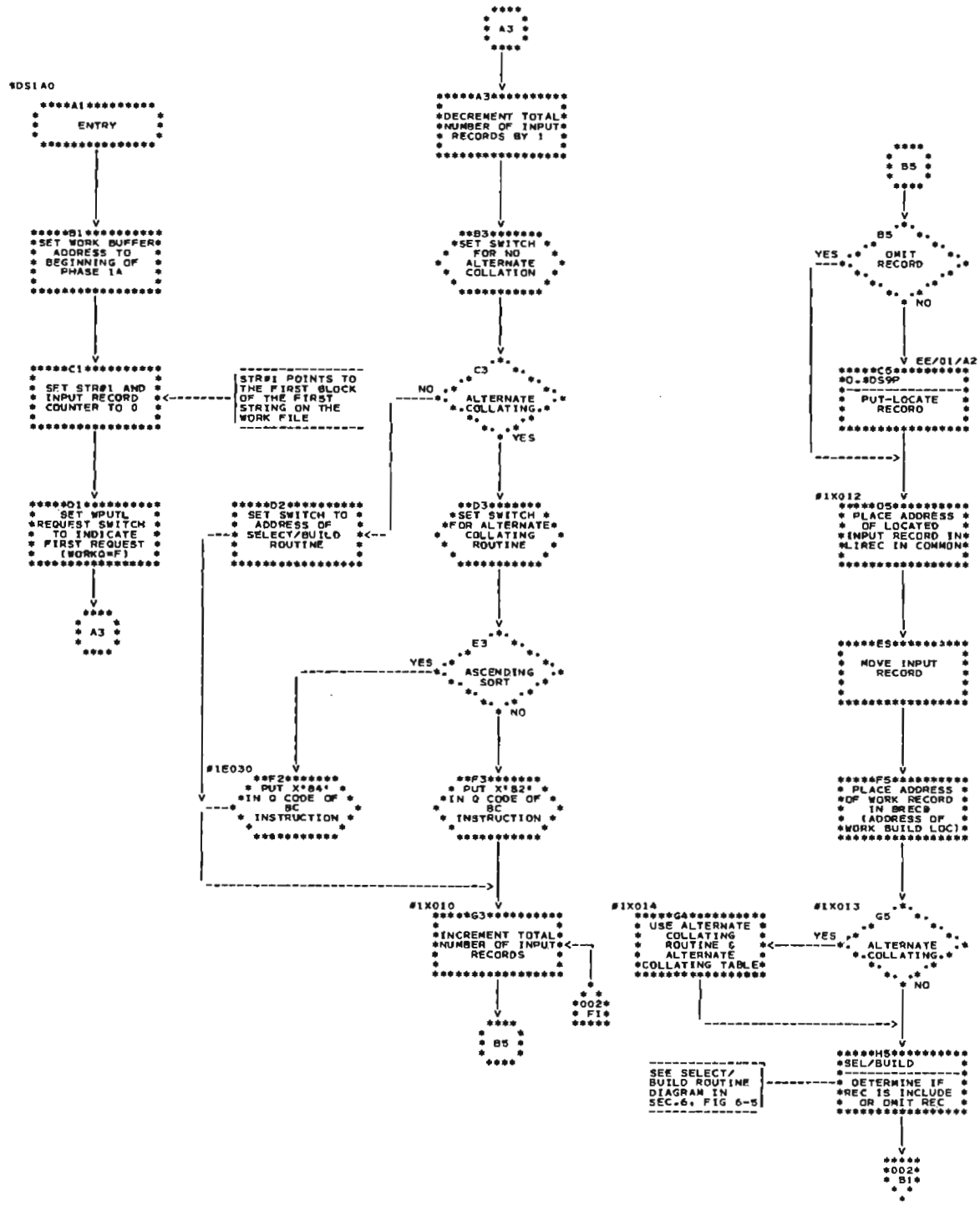


Chart AB (Part 1 of 3). Phase 1A – O.$DS1A – (Model 6, Model 10 Disk System)

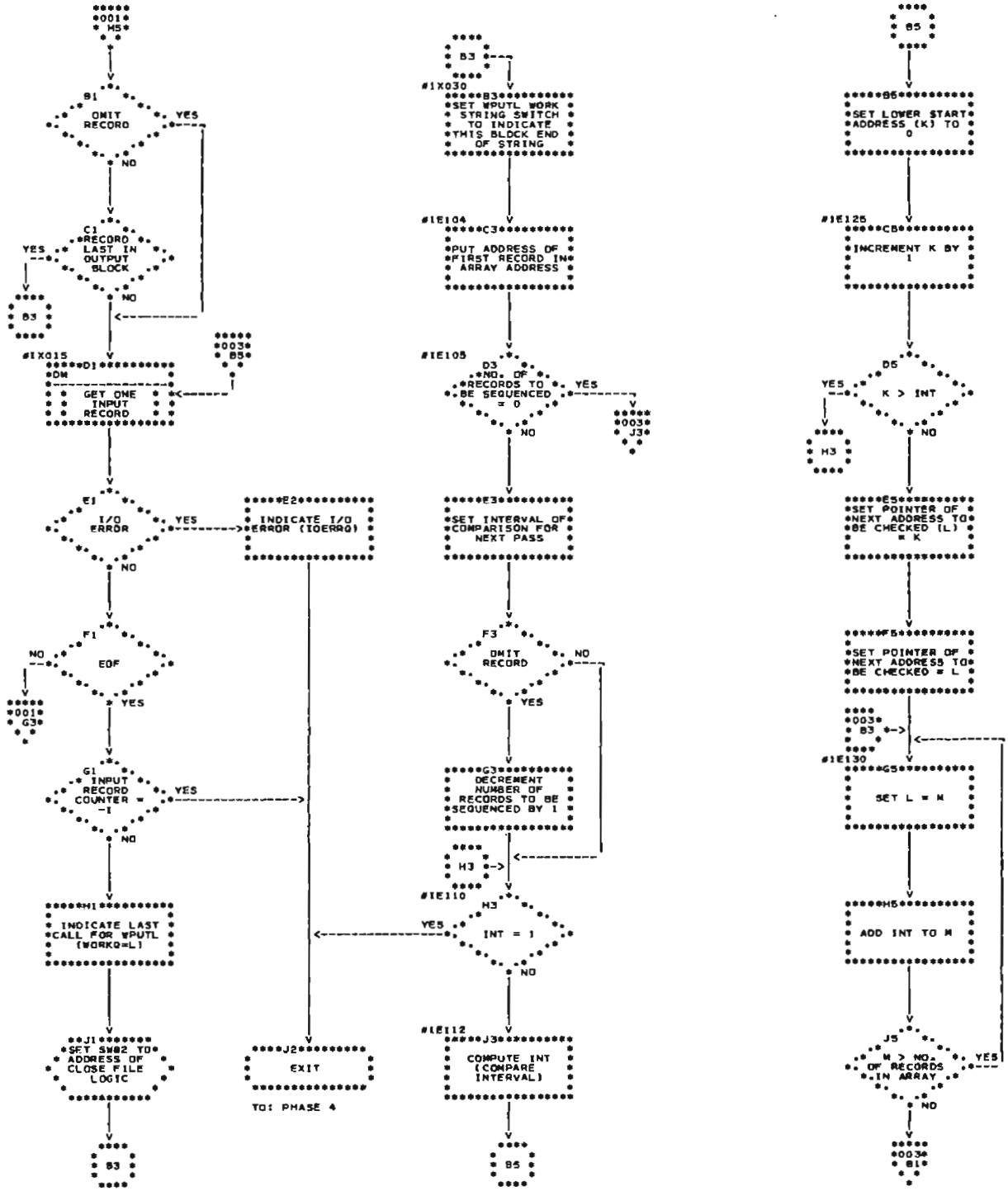


Chart AB (Part 2 of 3). Phase 1A – O.$DS1A – (Model 6, Model 10 Disk System)

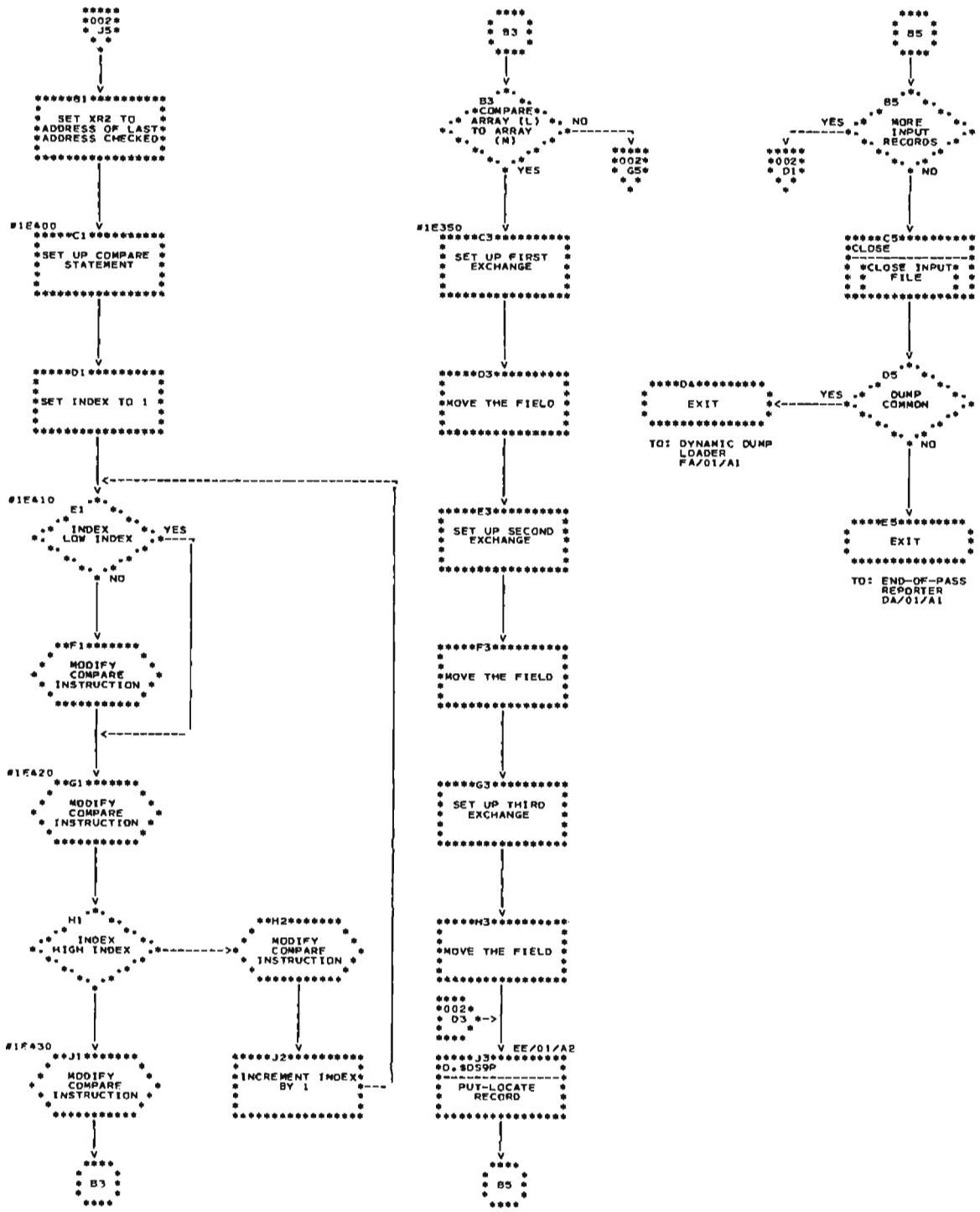


Chart AB (Part 3 of 3). Phase 1A – O.$DS1A – (Model 6, Model 10 Disk System)

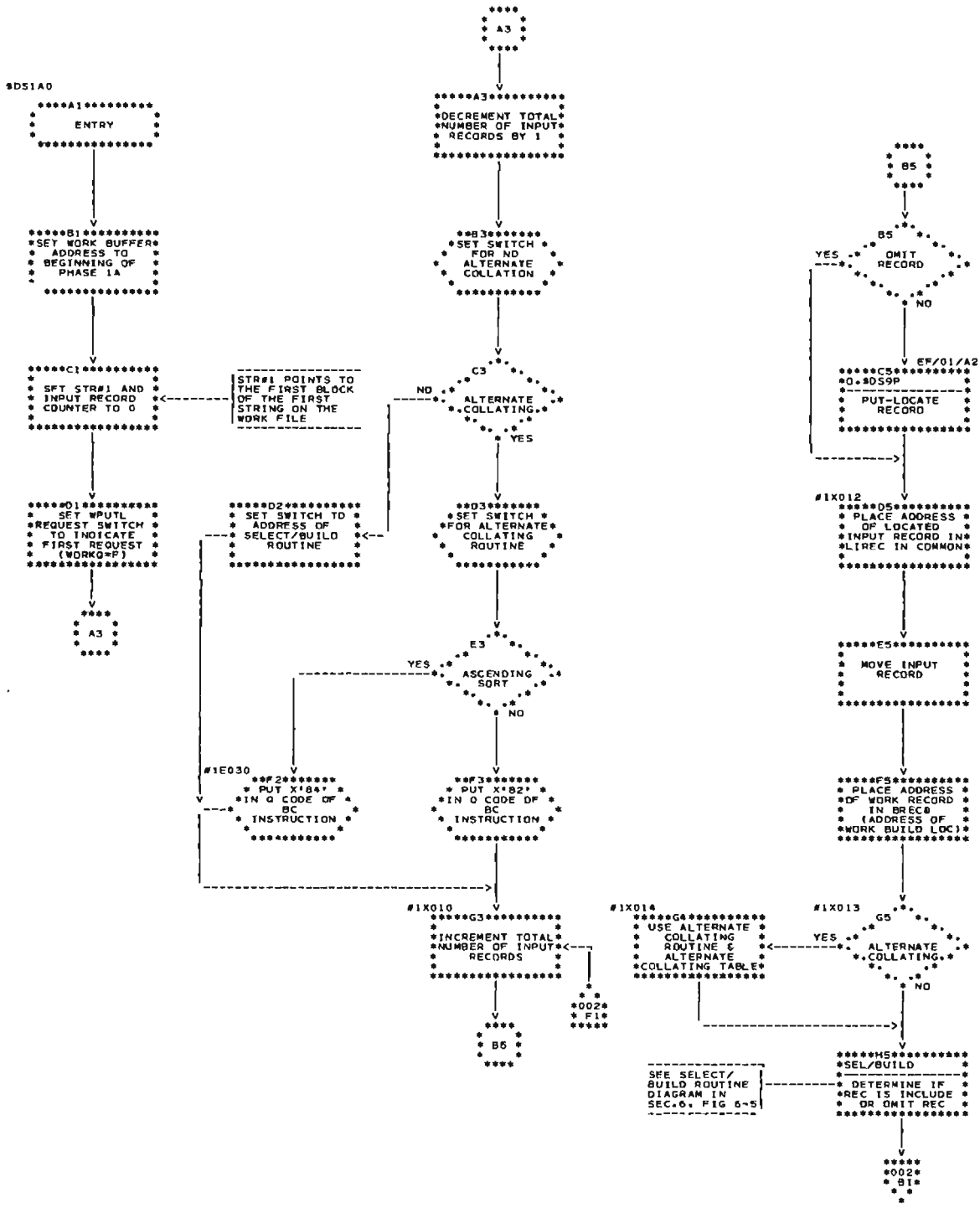
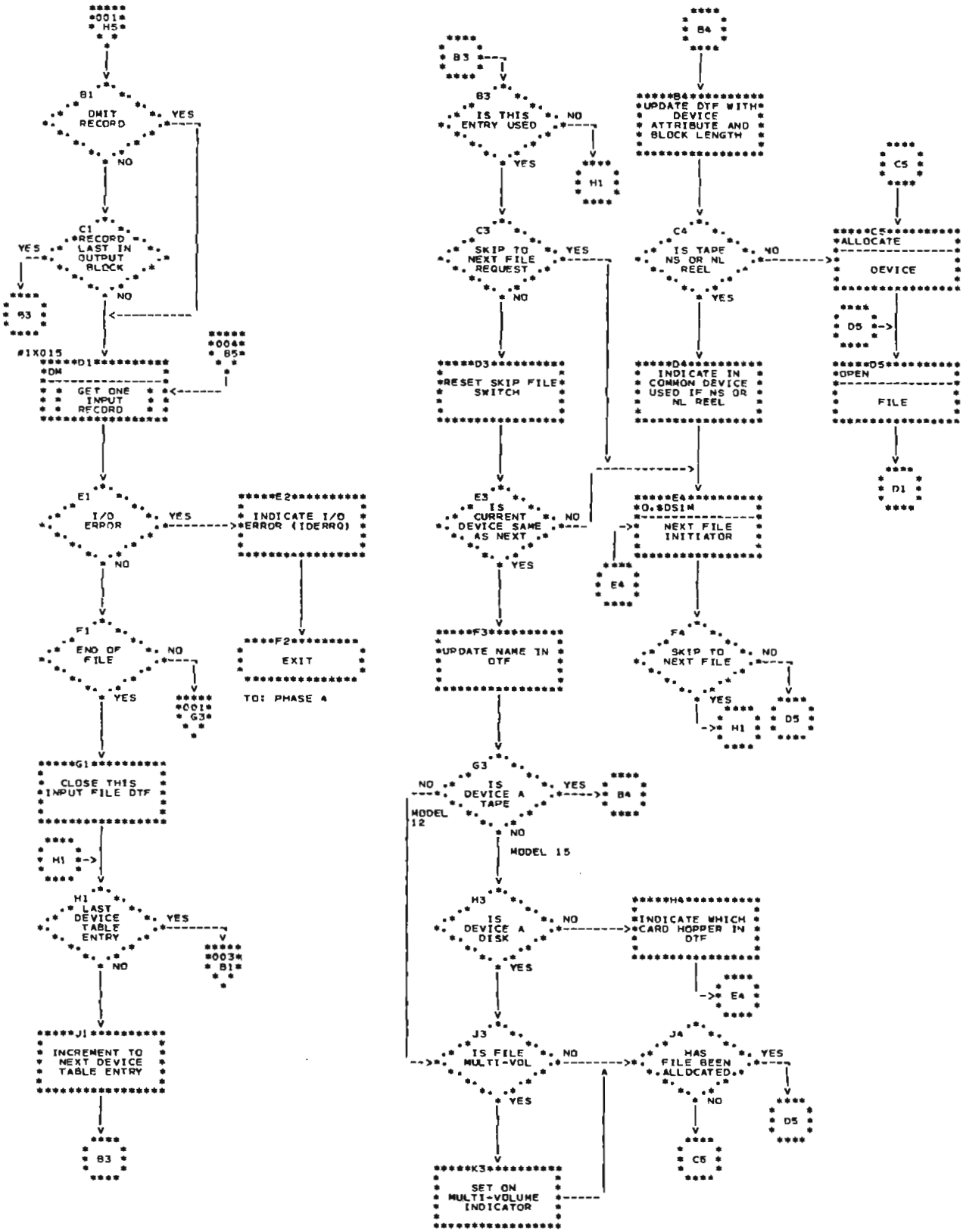


Chart AC (Part 1 of 4). Phase 1A -- O.$DS1A -- (Model 12, Model 15)



- Chart AC (Part 2 of 4). Phase 1A – O.$DS1A – (Model 12, Model 15)

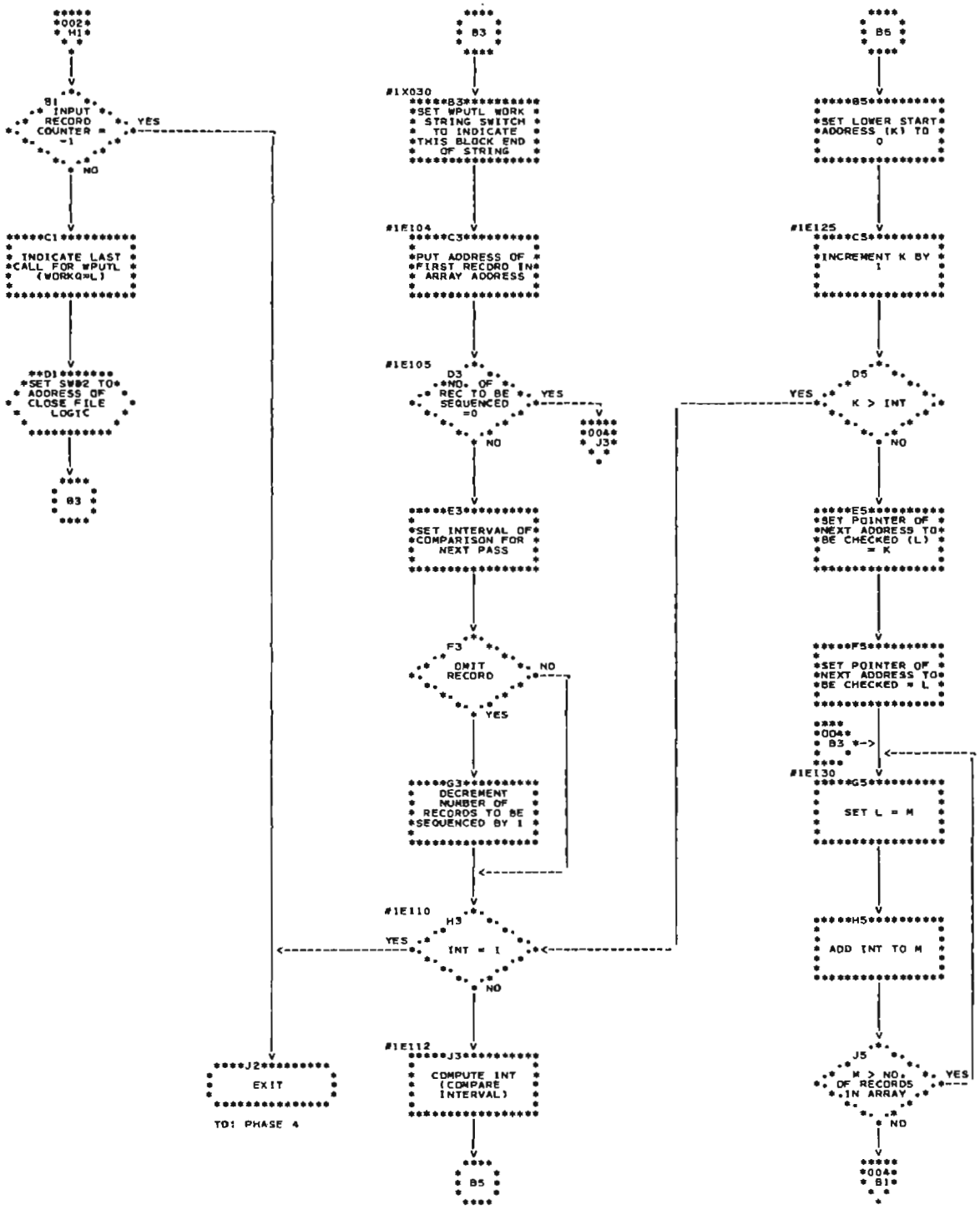


Chart AC (Part 3 of 4). Phase 1A – O.$DS1A – (Model 12, Model 15)

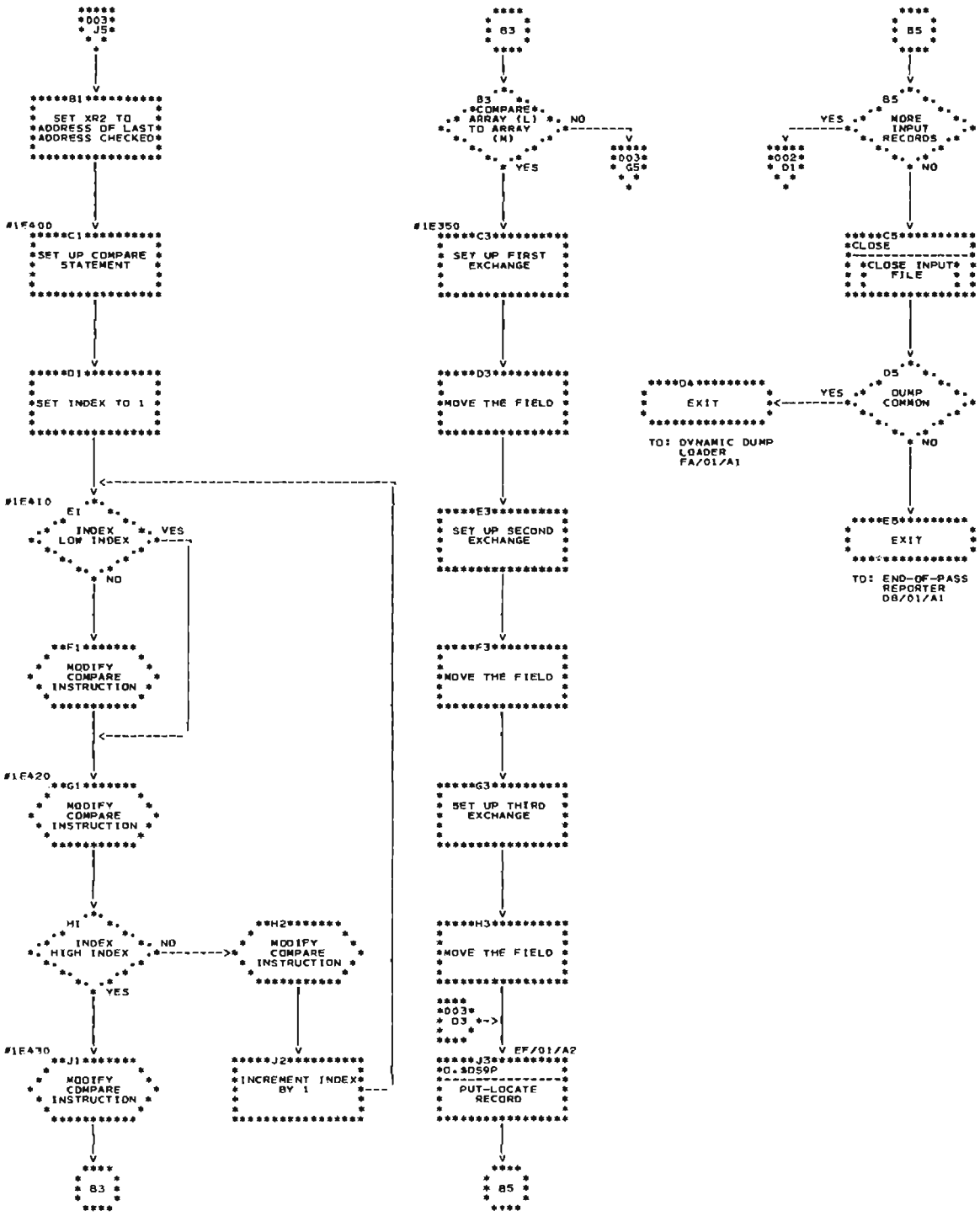


Chart AC (Part 4 of 4). Phase 1A – O.$DS1A – (Model 12, Model 15)

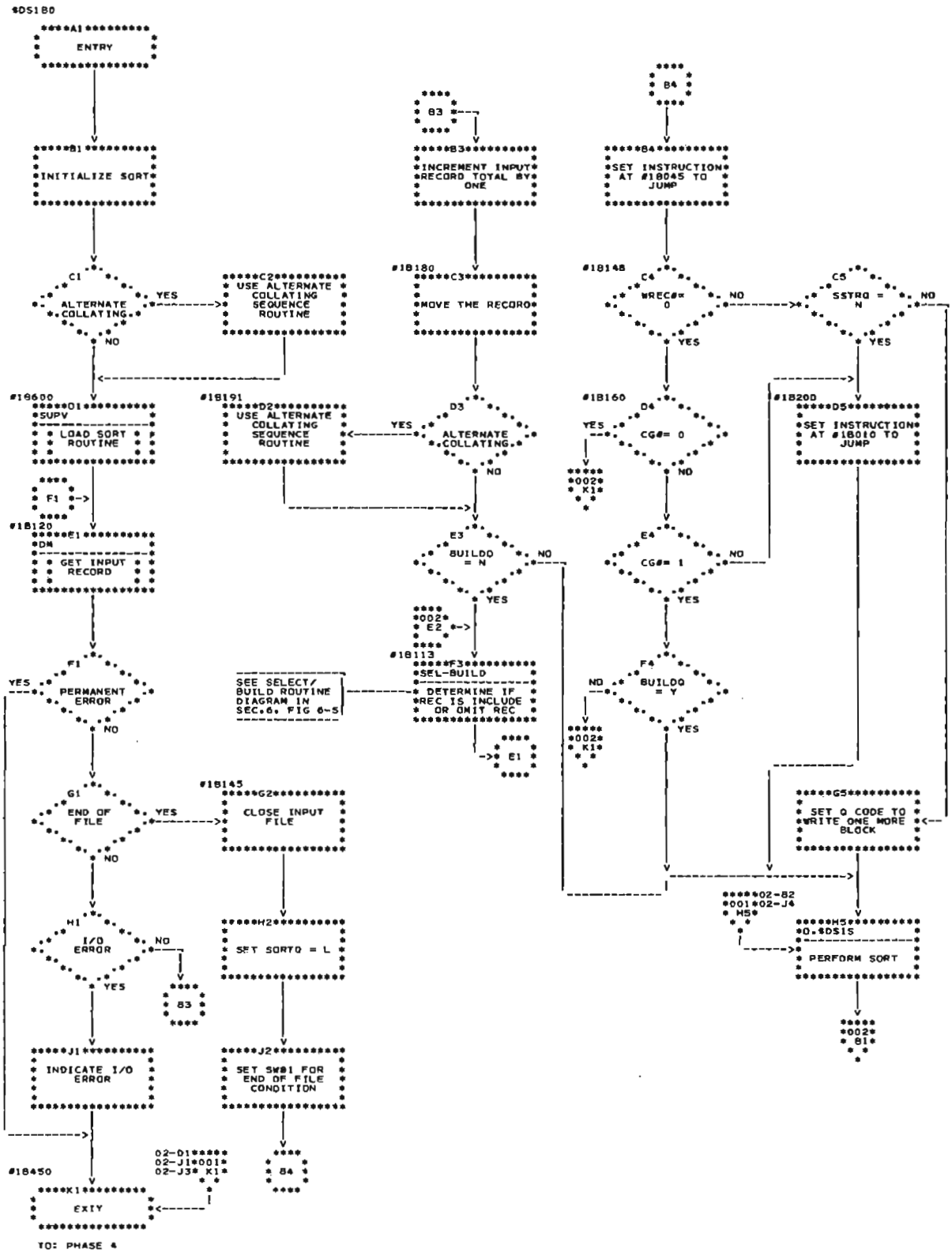


Chart AD (Part 1 of 2). Phase 1B – O.$DS1B – (Model 6, Model 10 Disk System)

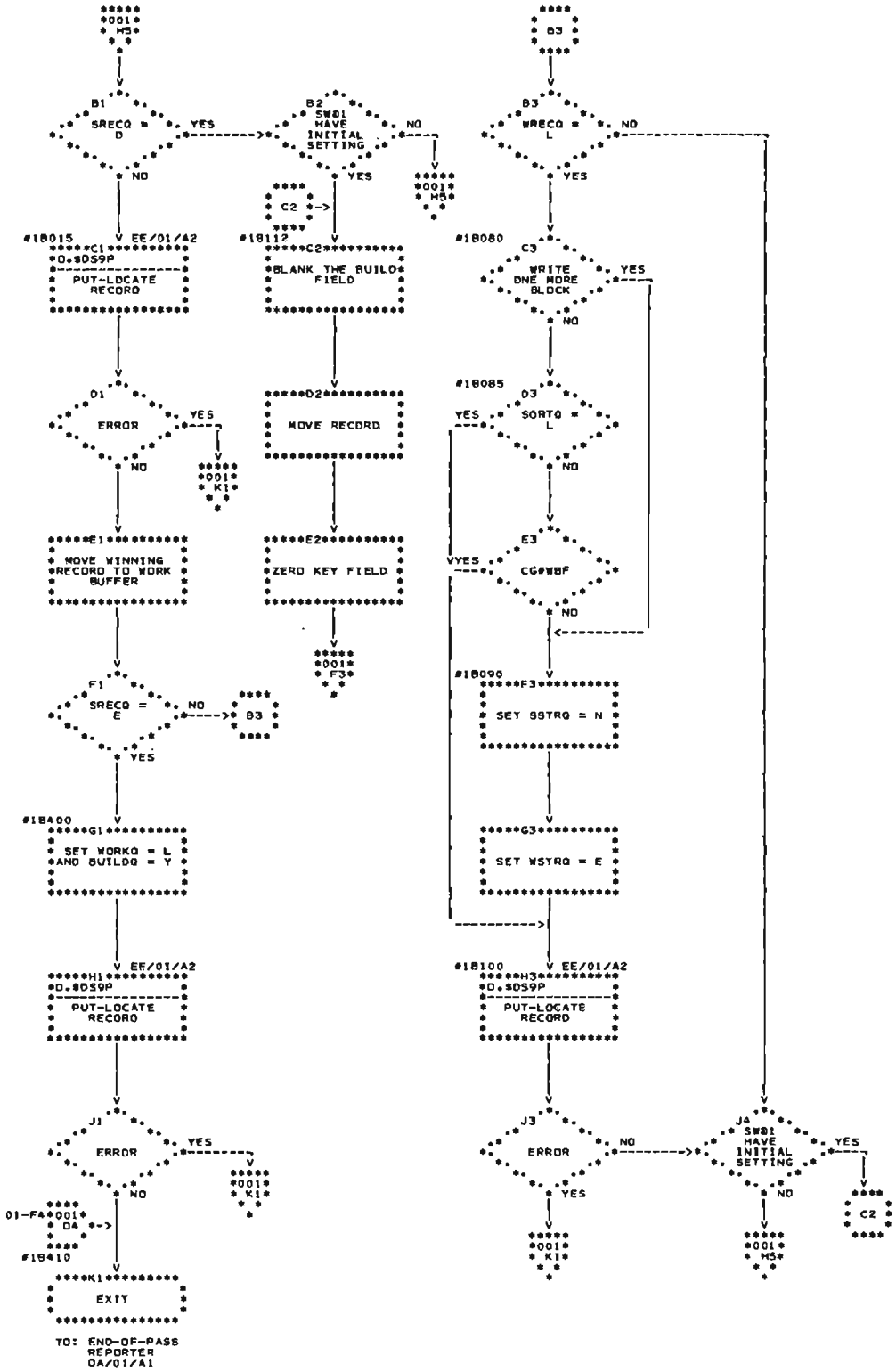
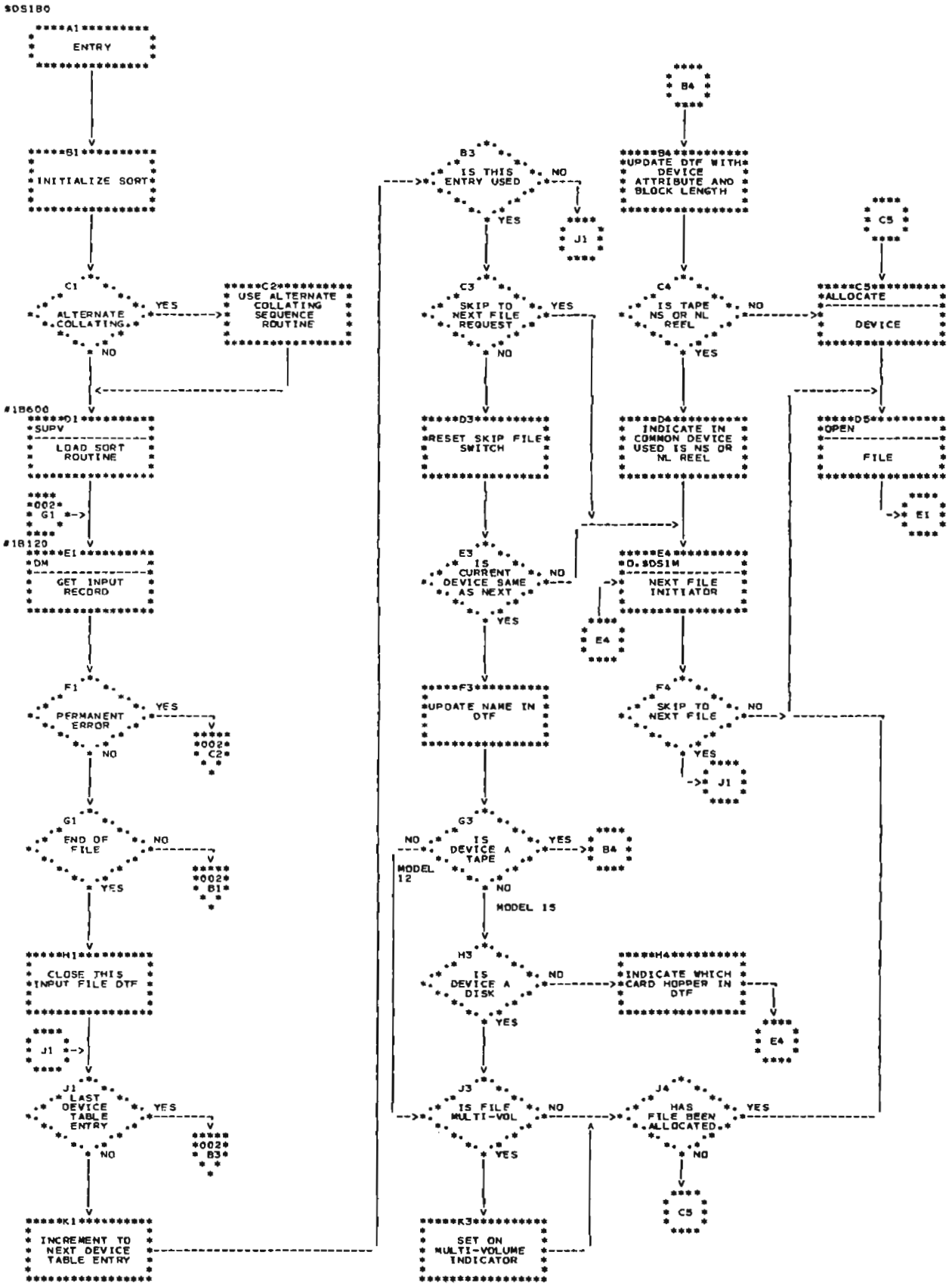


Chart AD (Part 2 of 2). Phase 1B – O.$DS1B – (Model 6, Model 10 Disk System)



- Chart AE (Part 1 of 3). Phase 1B – O.$DS1B – (Model 12, Model 15)

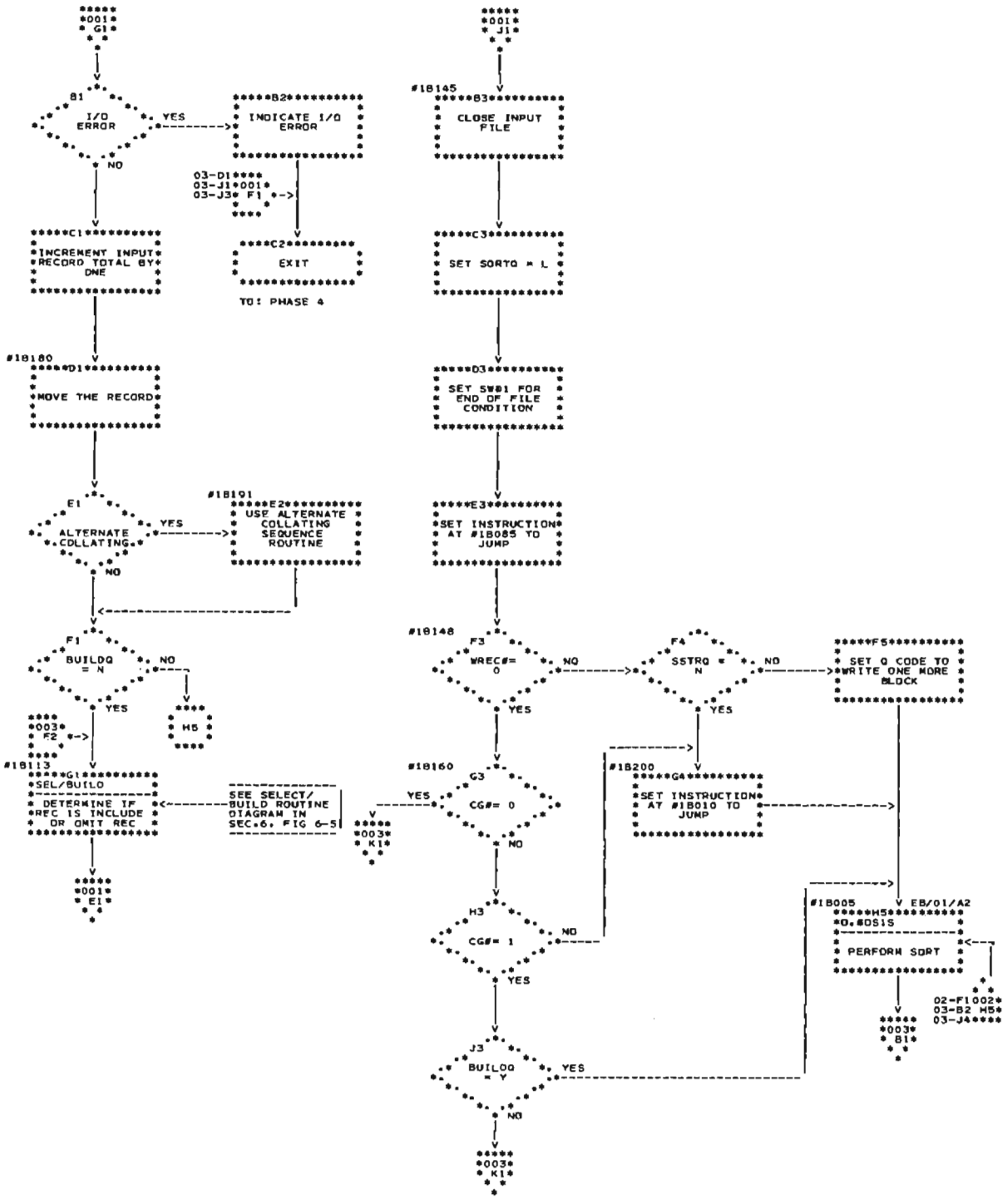


Chart AE (Part 2 of 3). Phase 1B – O.$DS1B – (Model 12, Model 15)

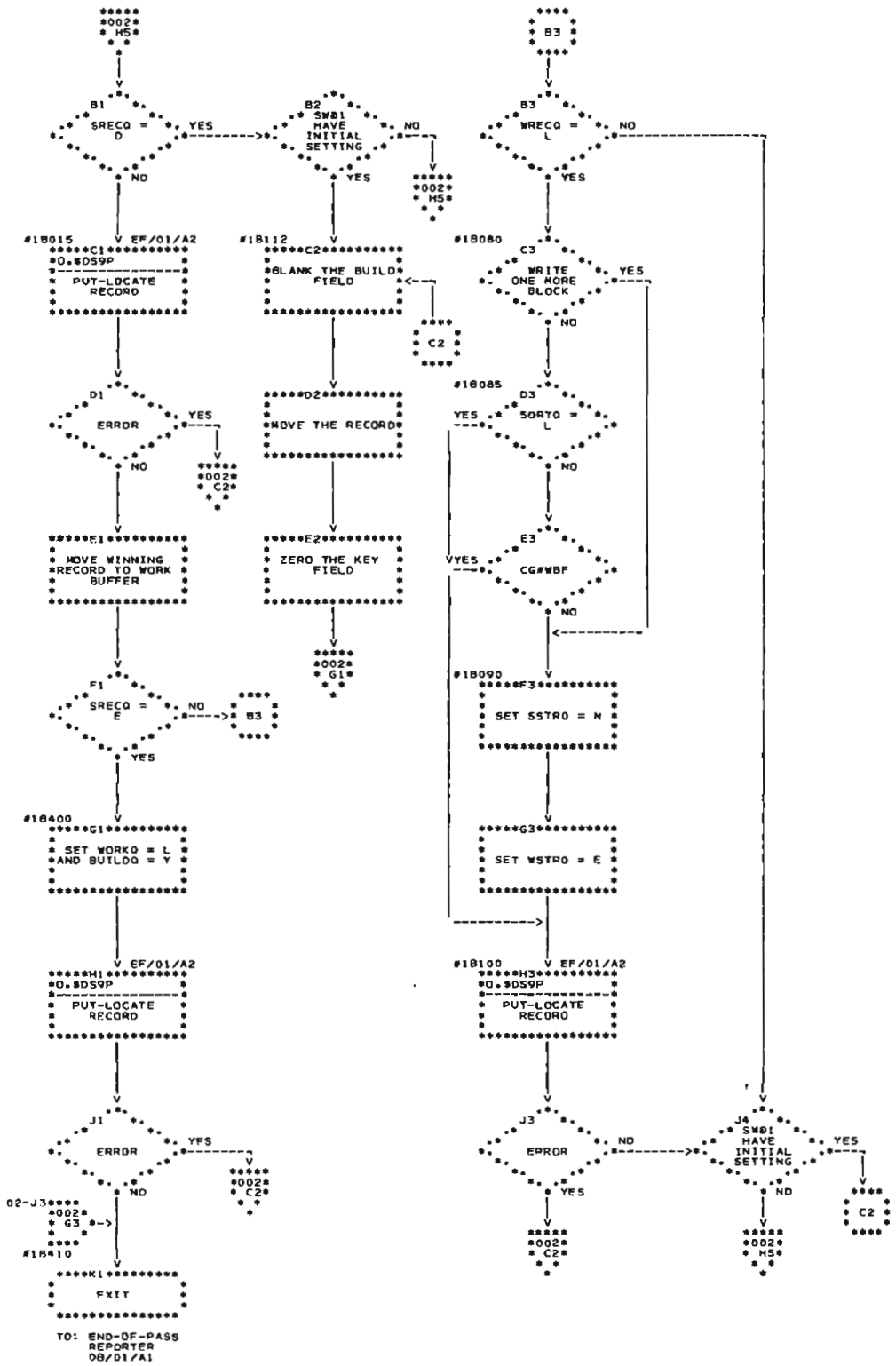


Chart AE (Part 3 of 3). Phase 1B – O.$DS1B – (Model 12, Model 15)

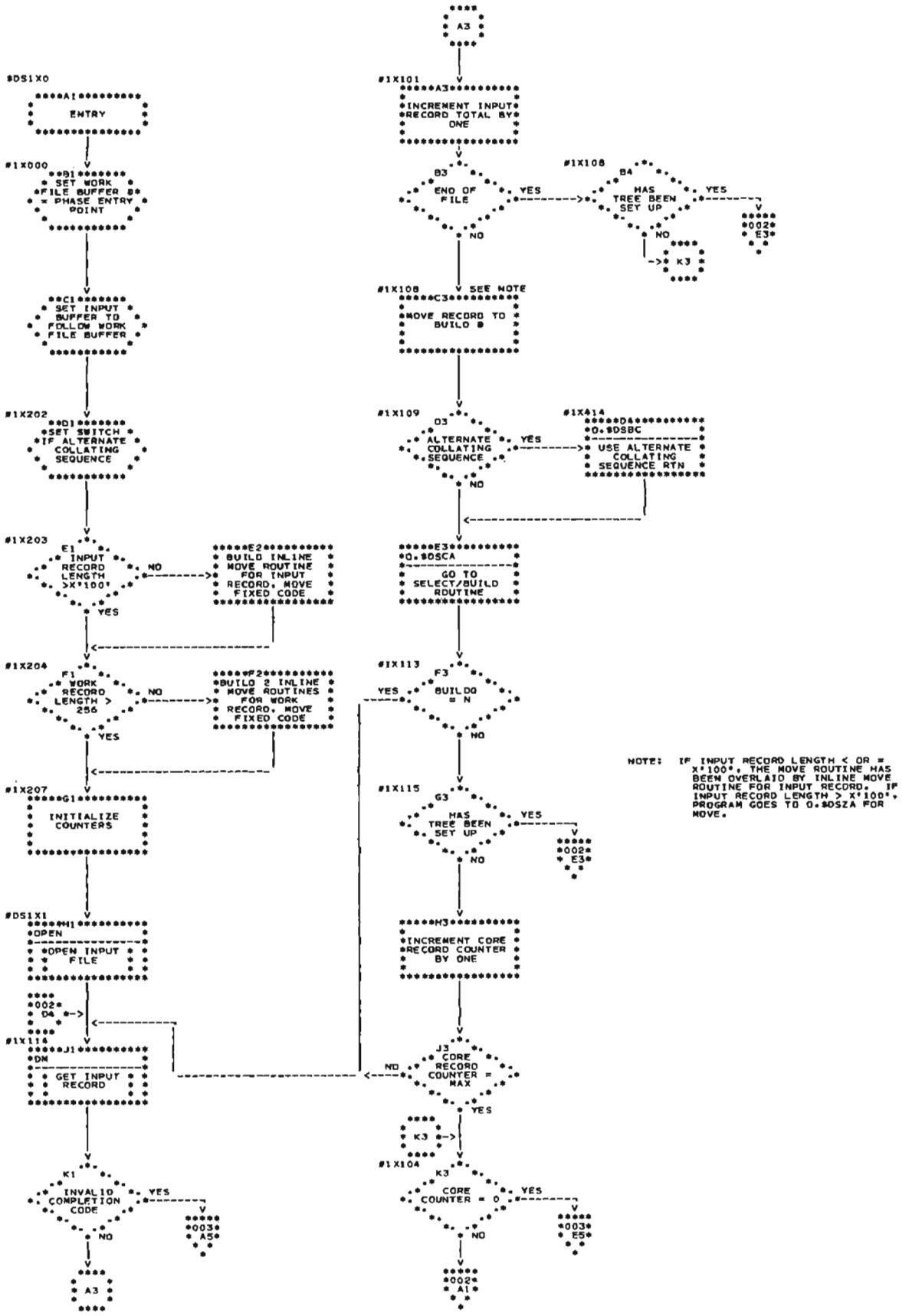


Chart AF (Part 1 of 3). Phase 1X – O.$DS1X – (Model 6, Model 10 Disk System)

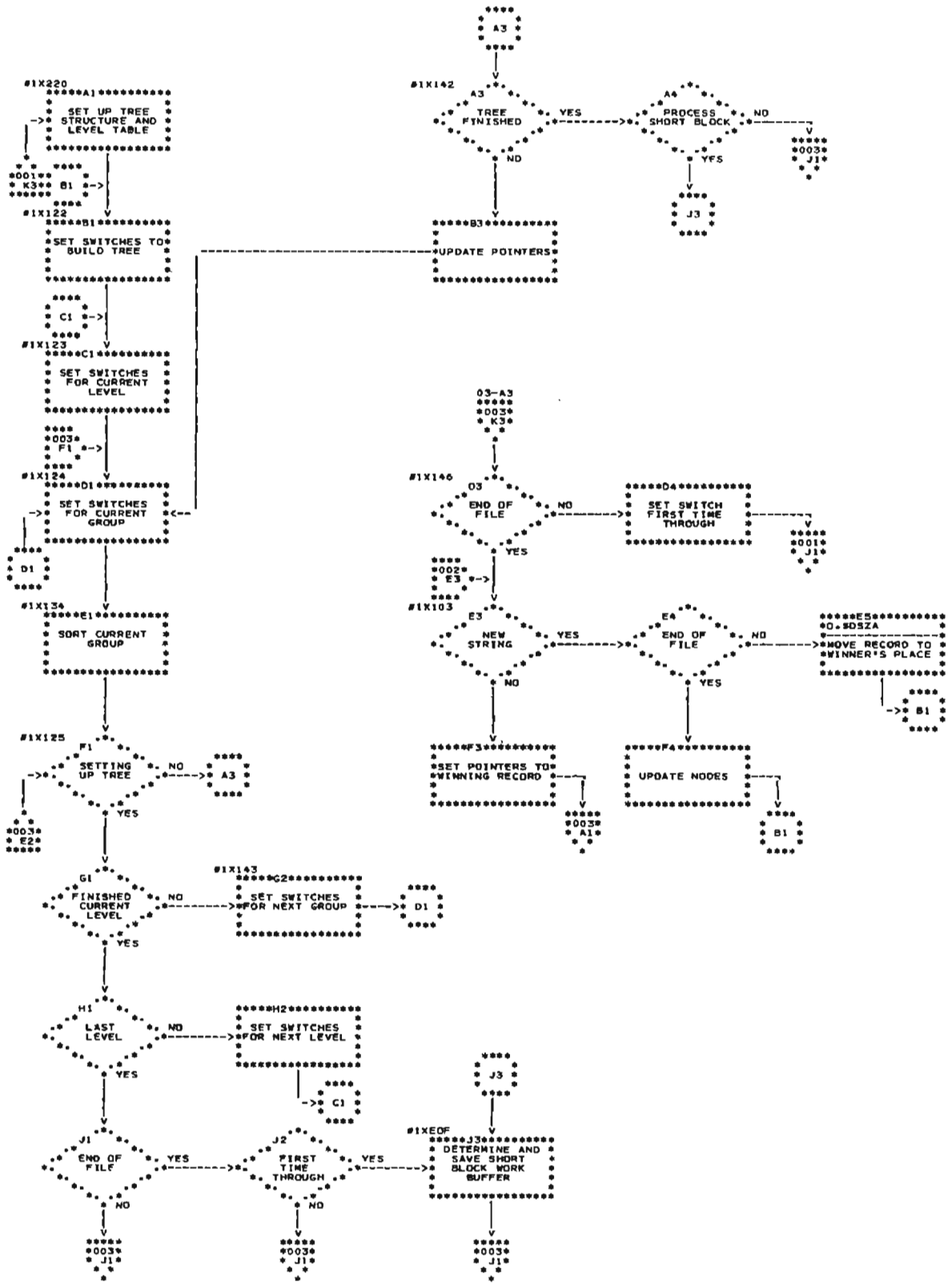


Chart AF (Part 2 of 3). Phase 1X – O.$DS1X – (Model 6, Model 10 Disk System)

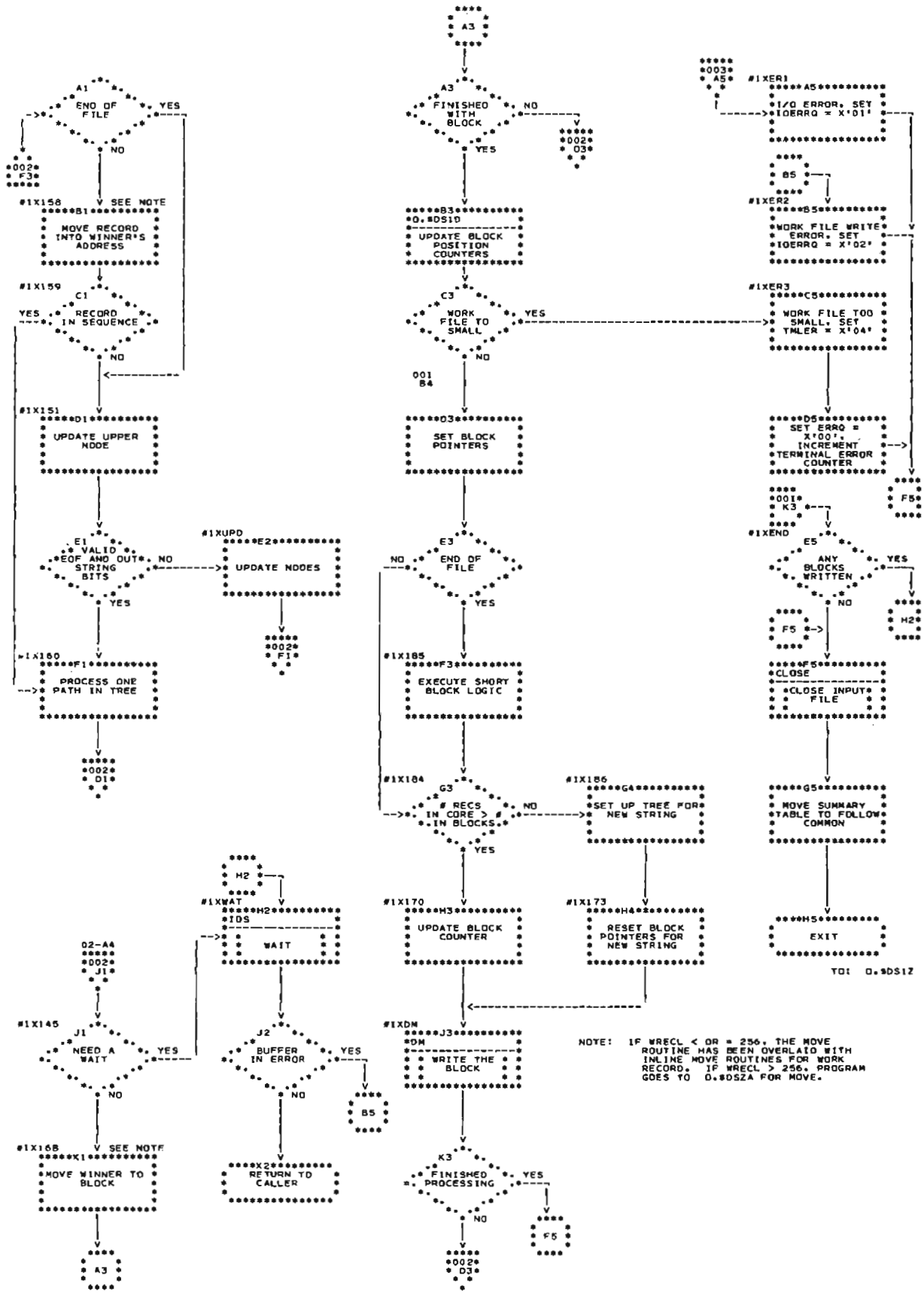
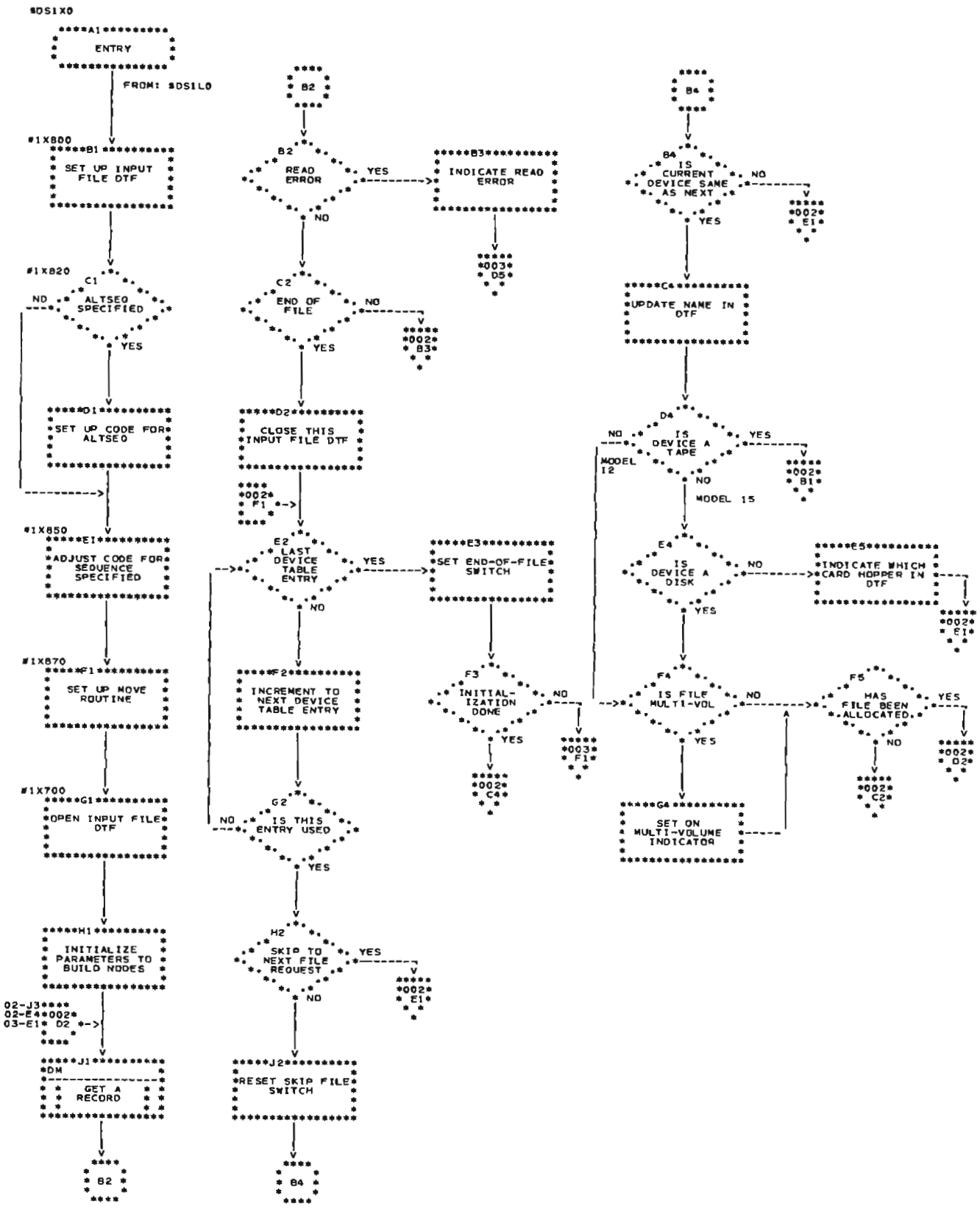


Chart AF (Part 3 of 3). Phase 1X – O.$DS1X – (Model 6, Model 10 Disk System)



- Chart AG (Part 1 of 3). Phase 1X – O.$DS1X – (Model 12, Model 15)

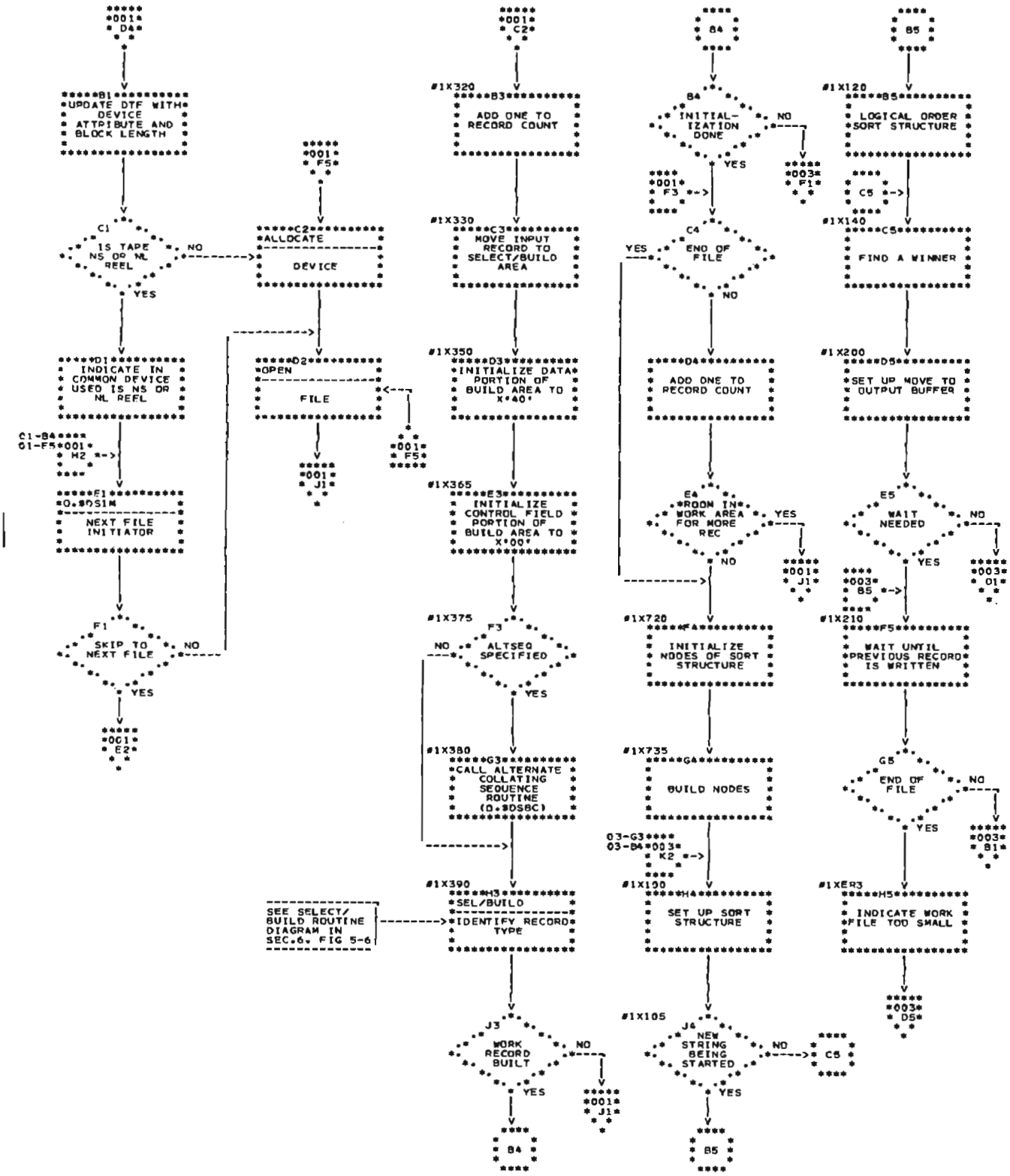


Chart AG (Part 2 of 3). Phase 1X – O.$DS1X – (Model 12, Model 15)

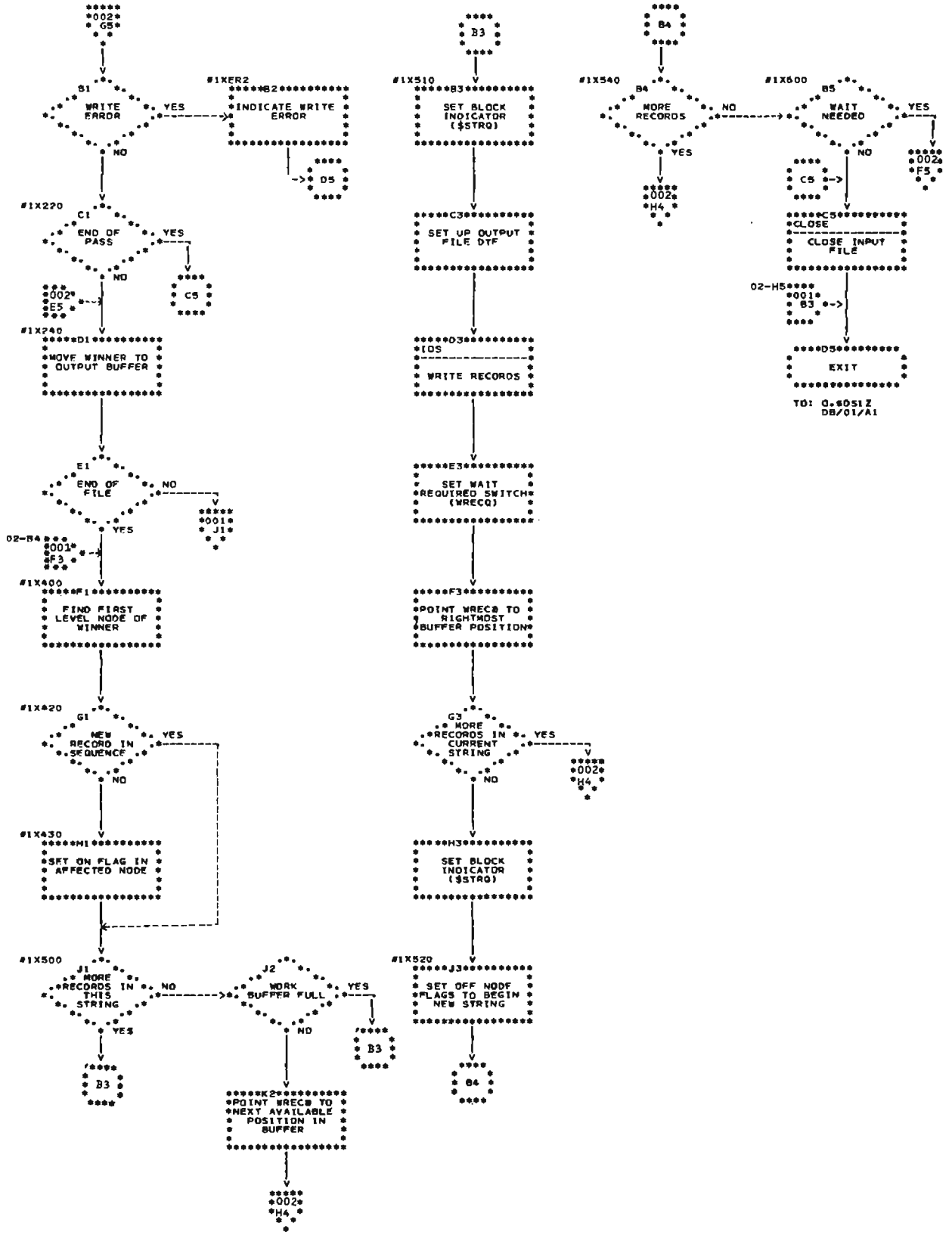


Chart AG (Part 3 of 3). Phase 1X – O.$DS1X – (Model 12, Model 15)

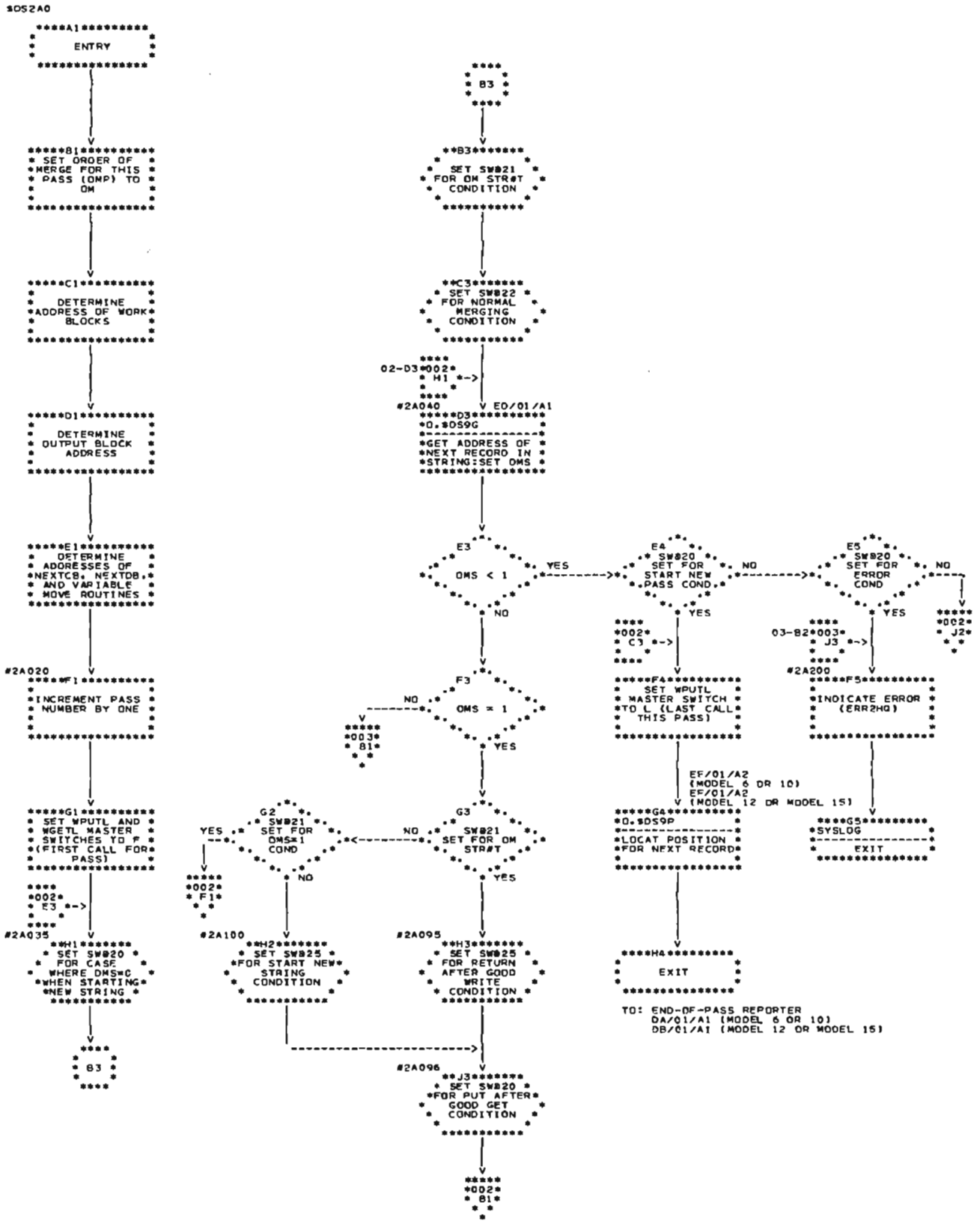


Chart BA (Part 1 of 3). Phase 2A – O.$DS2A – (All Models)

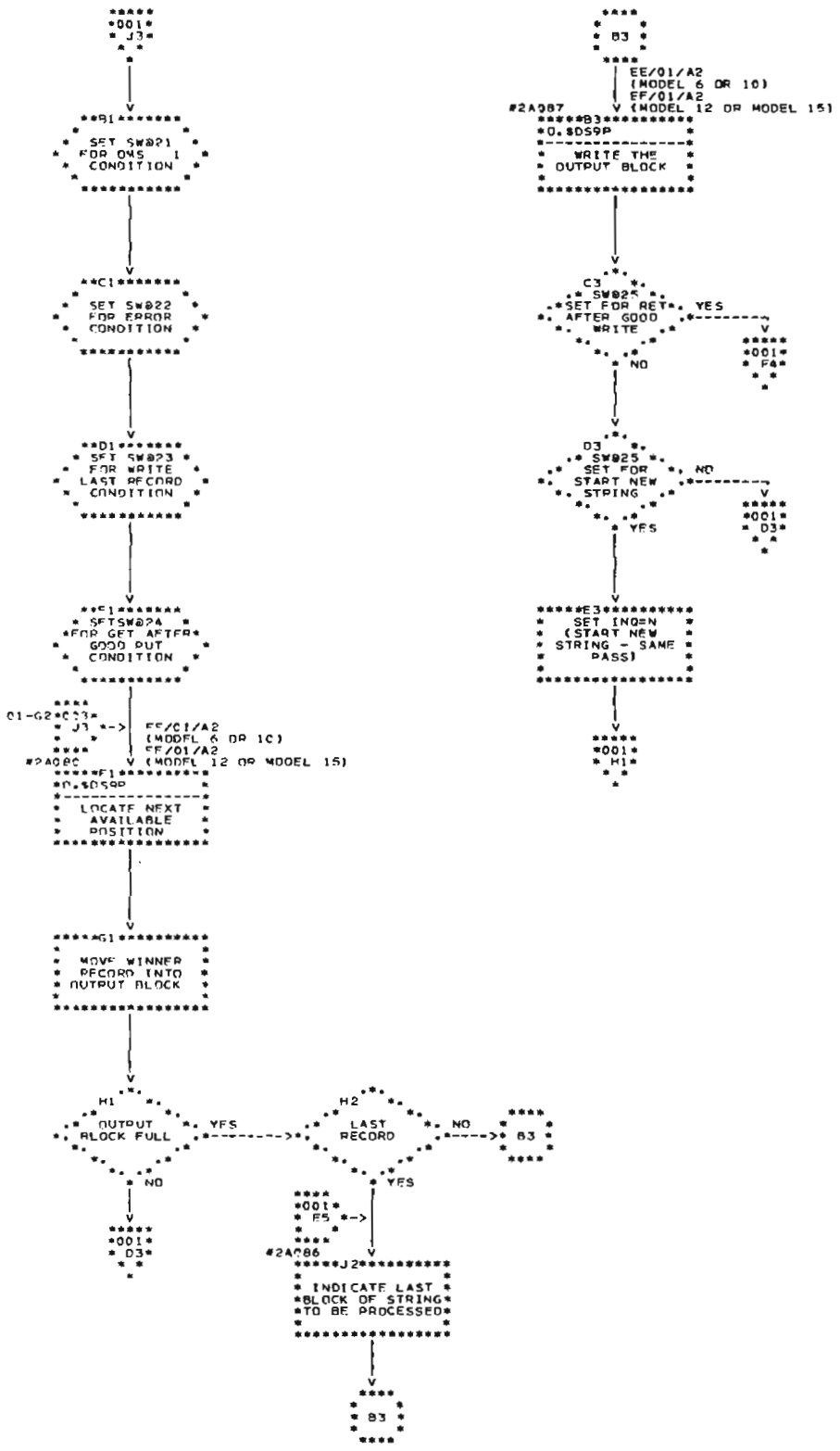


Chart BA (Part 2 of 3). Phase 2A – O.$DS2A – (All Models)

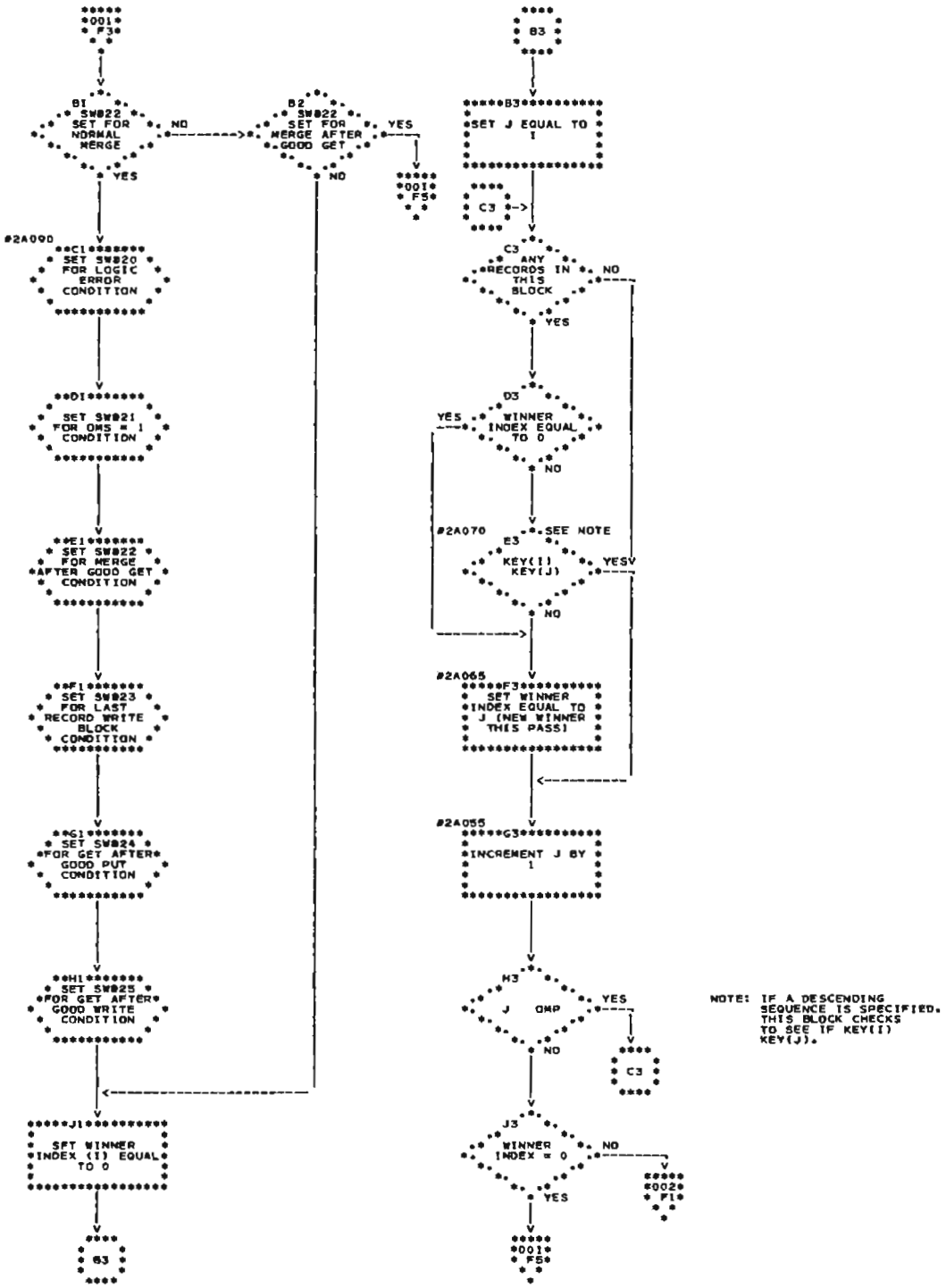


Chart BA (Part 3 of 3). Phase 2A – O.$DS2A – (All Models)

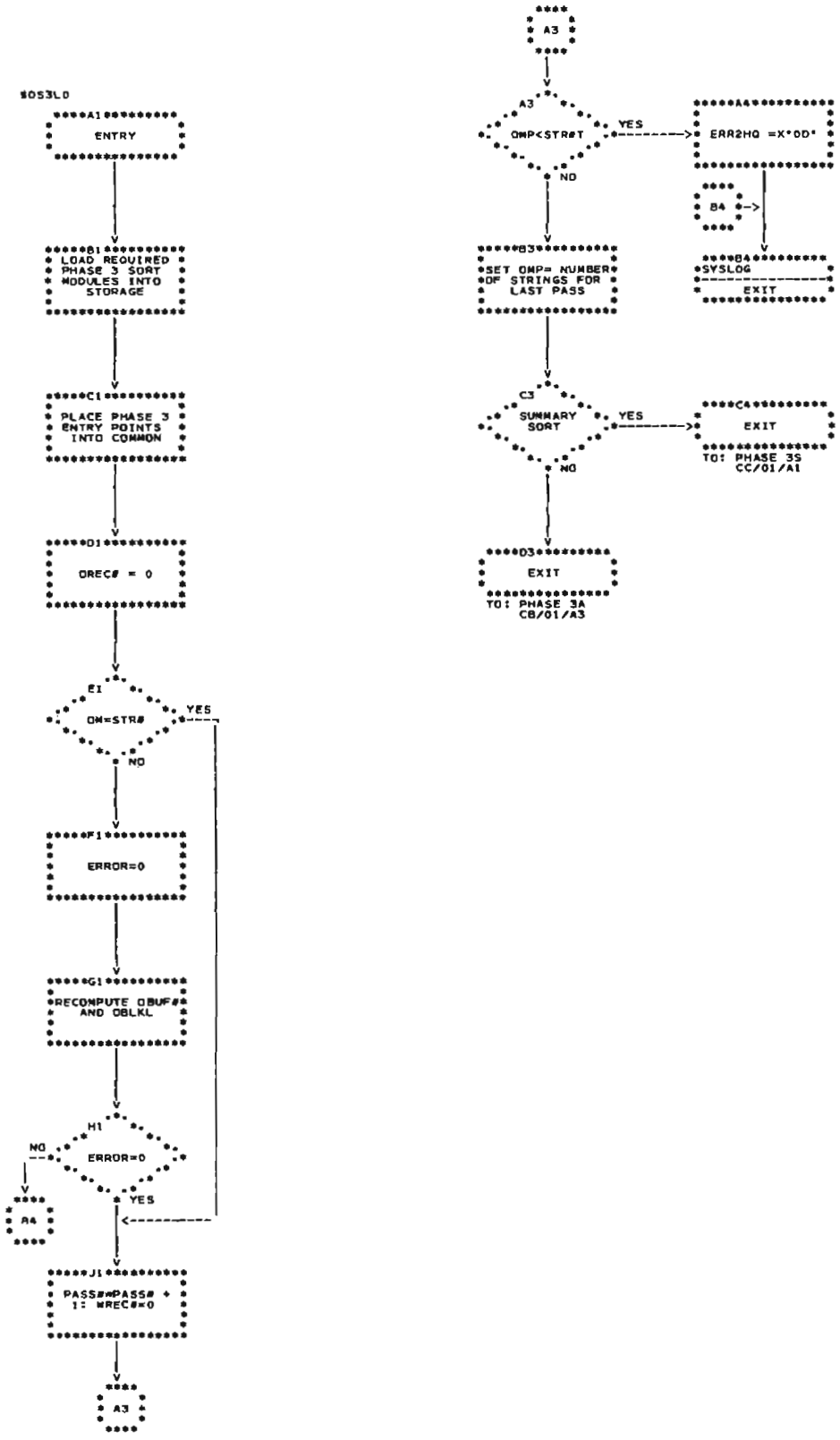


Chart CA. Phase 3L – O.$DS3L – (All Models)

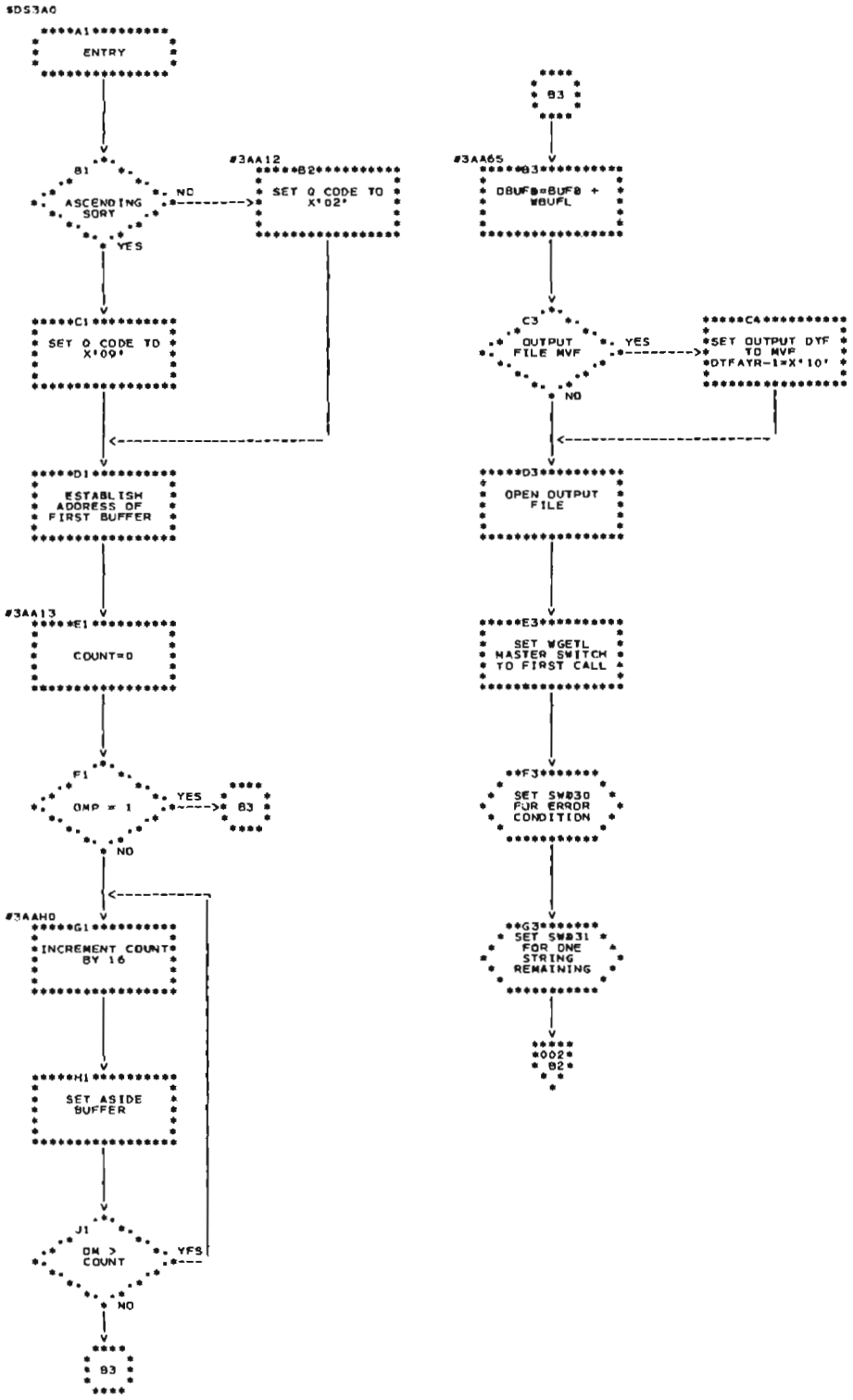


Chart CB (Part 1 of 3). Phase 3A – O.$DS3A – (All Models)

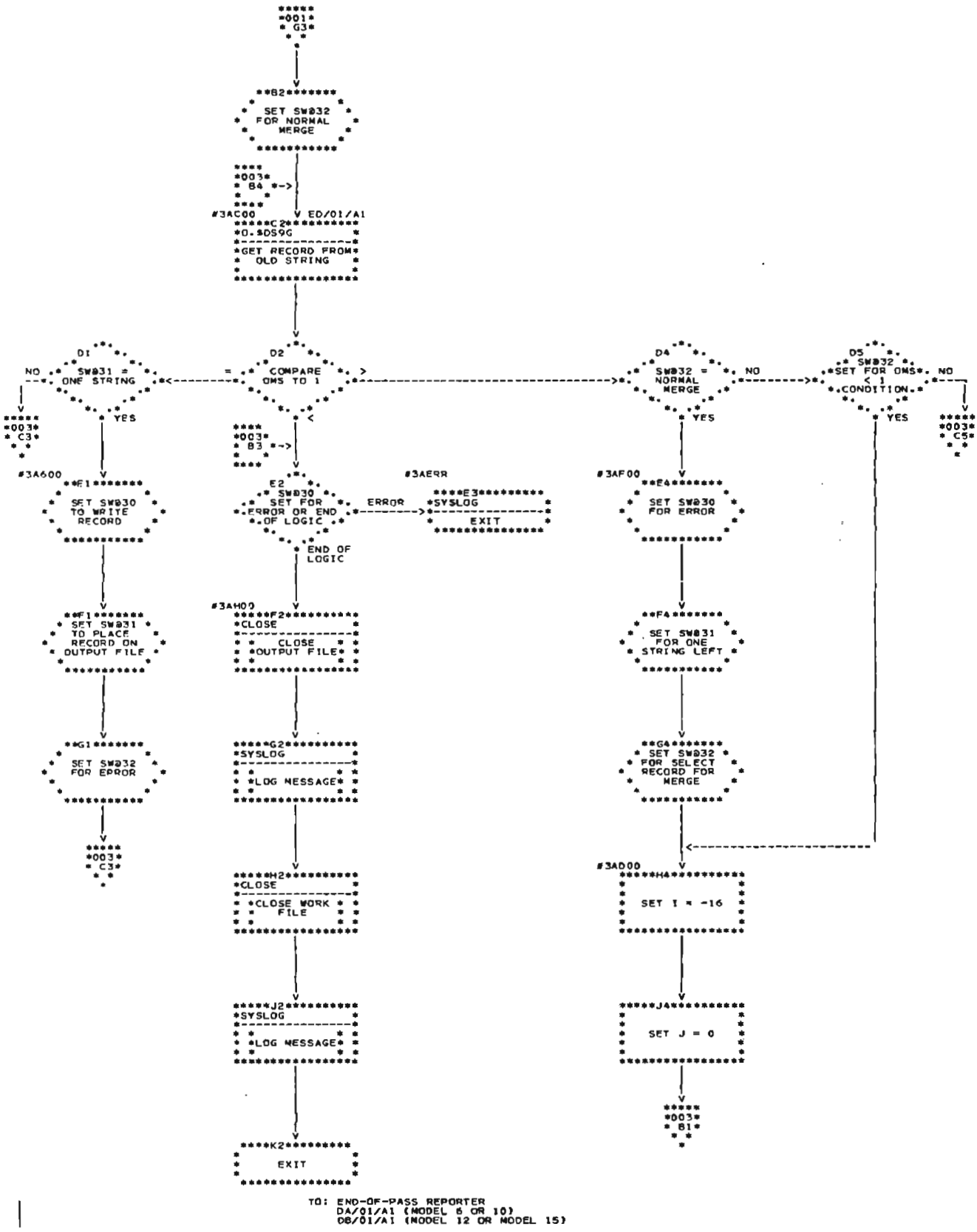


Chart CB (Part 2 of 3). Phase 3A – O.$DS3A – (All Models)

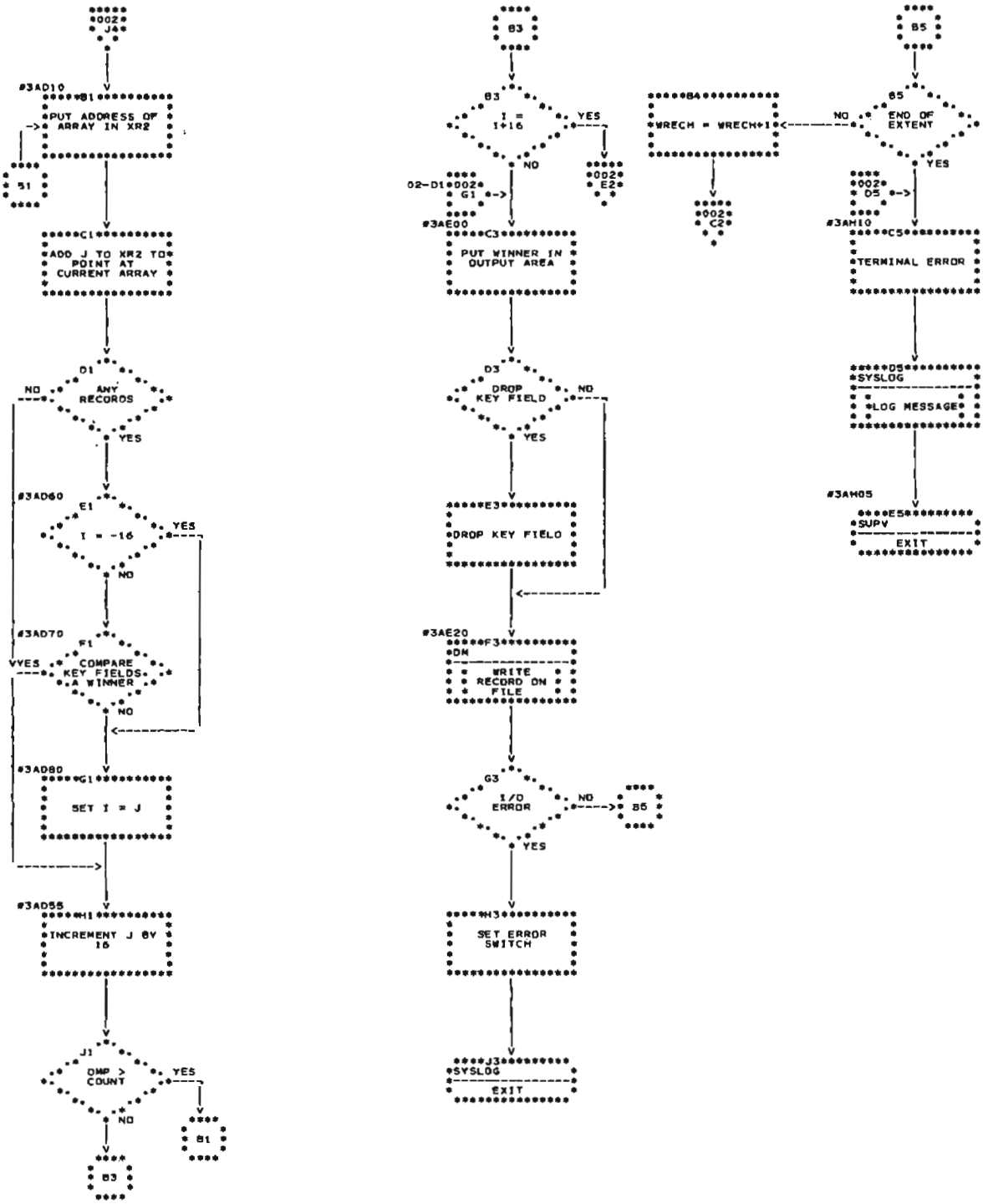


Chart CB (Part 3 of 3). Phase 3A – O.$DS3A – (All Models)

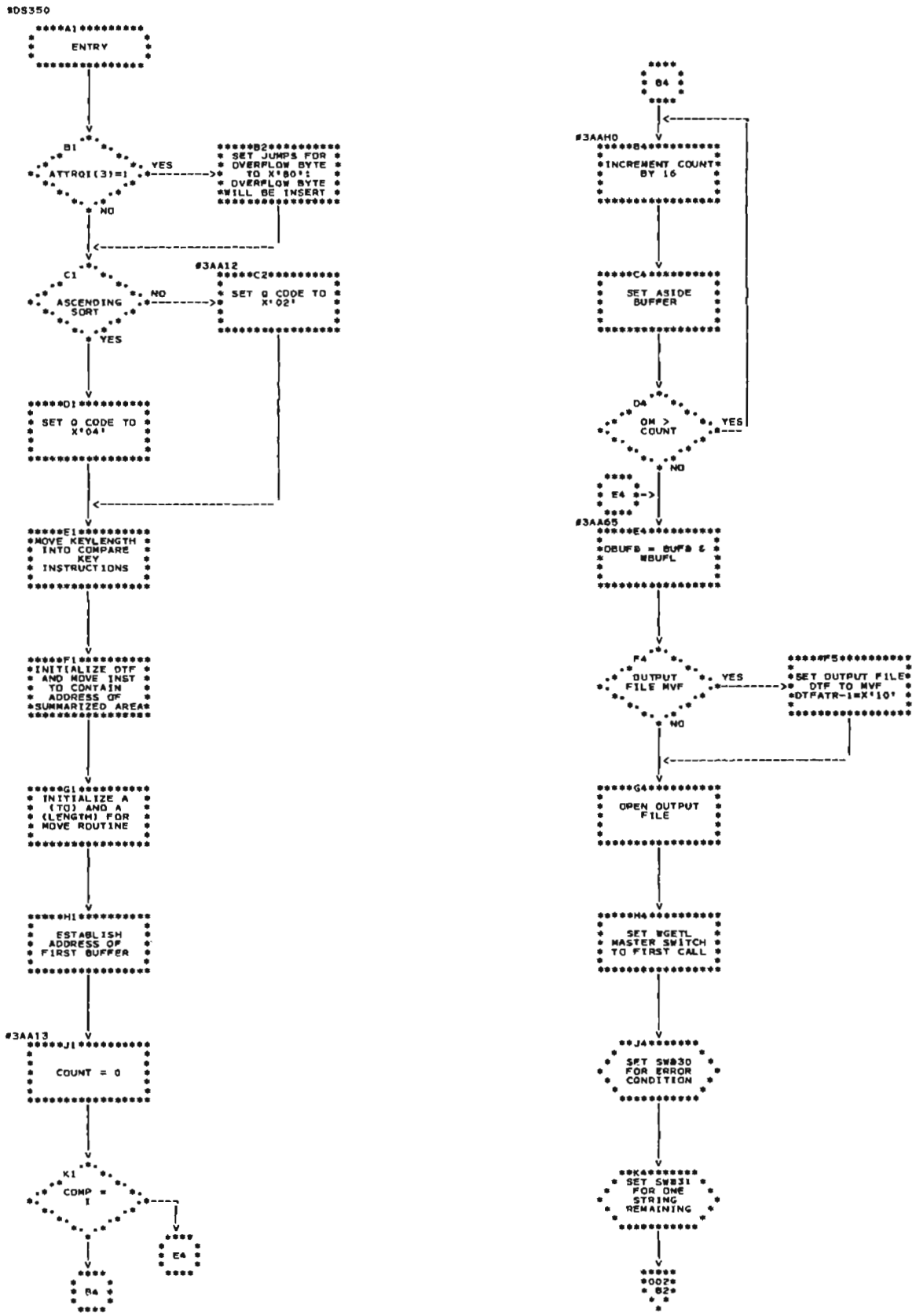


Chart CC (Part 1 of 4). Phase 3S – O.$DS3S – (All Models)

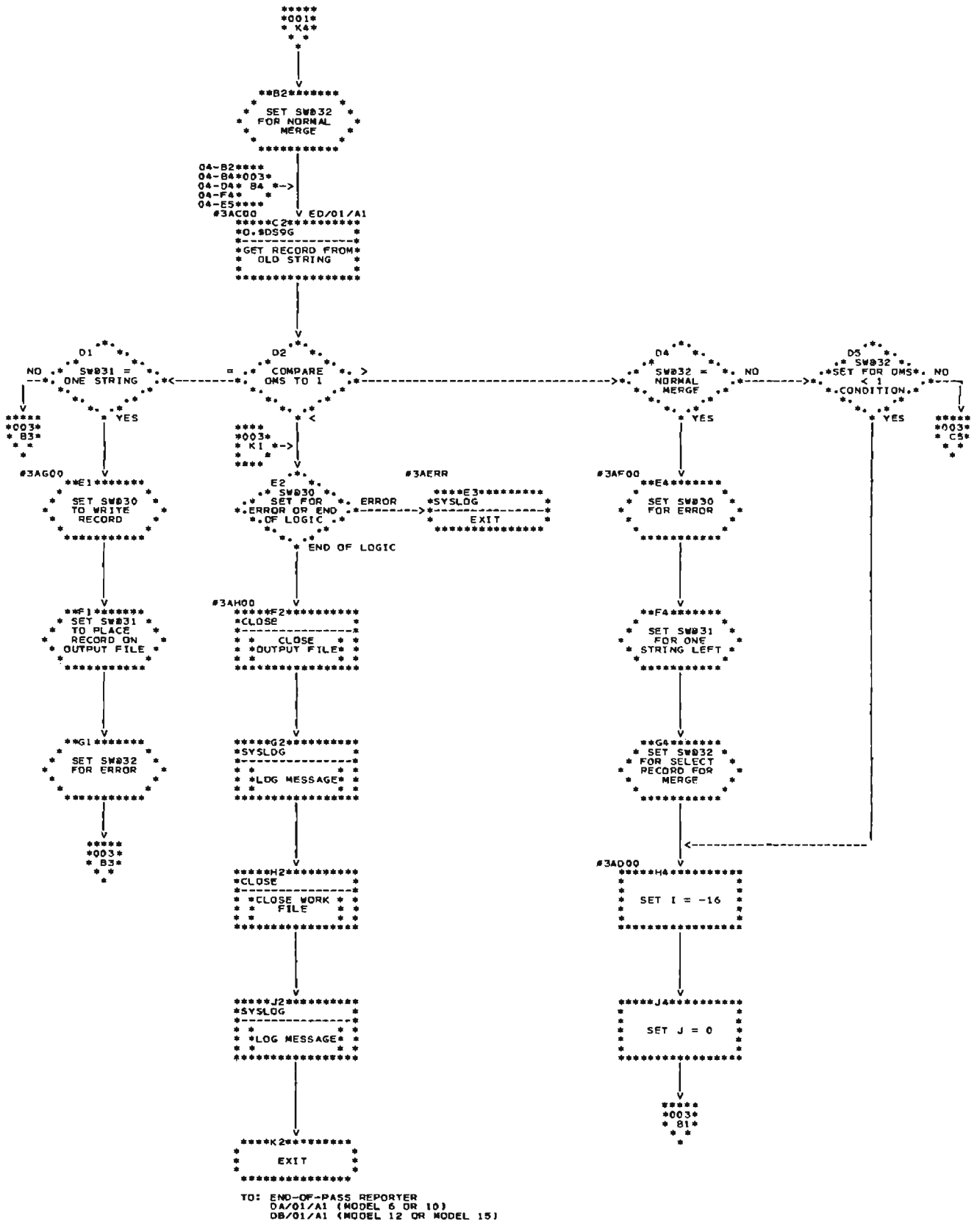


Chart CC (Part 2 of 4). Phase 3S – O.$DS3S – (All Models)

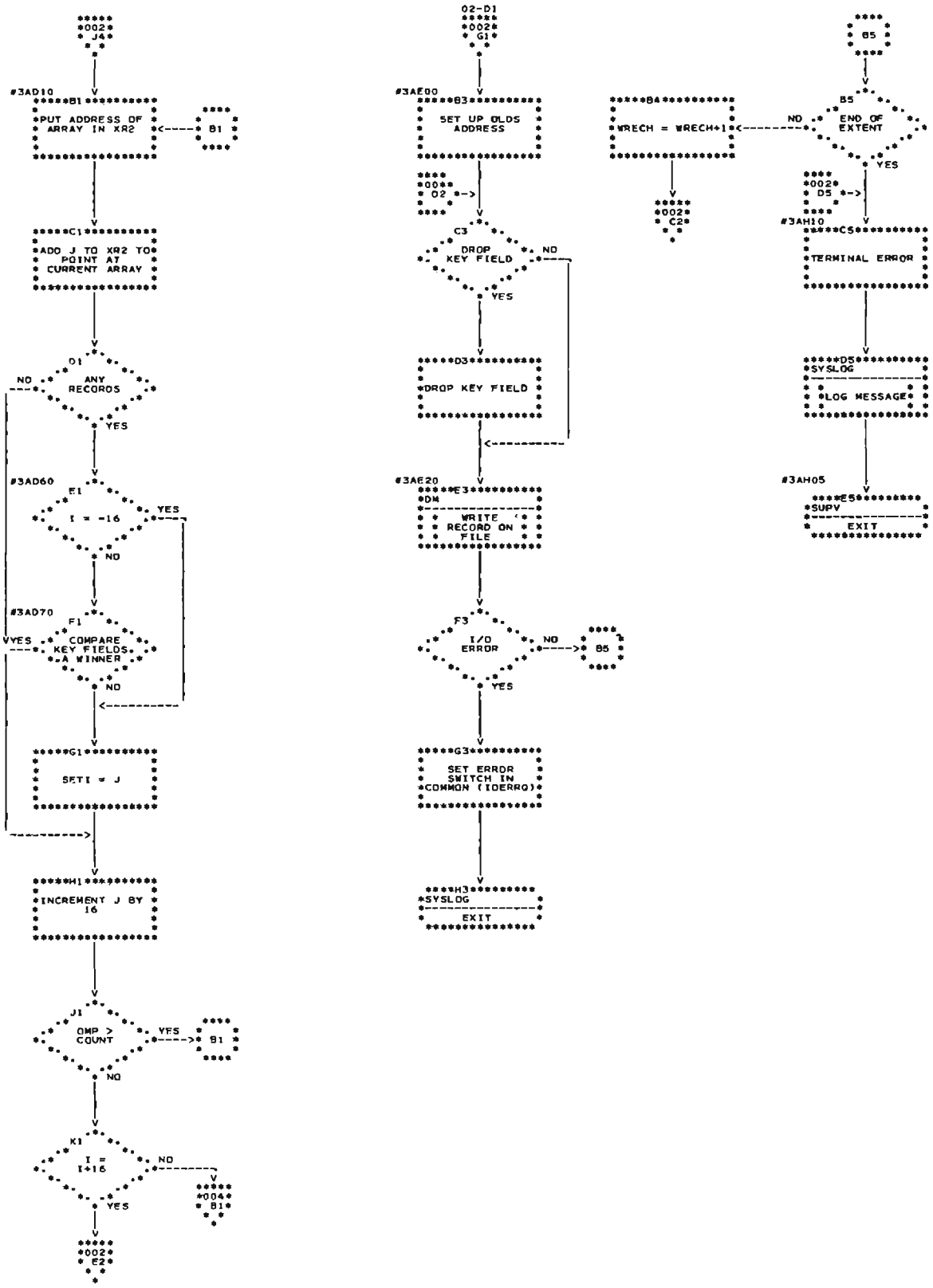


Chart CC (Part 3 of 4). Phase 3S – O.$DS3S – (All Models)

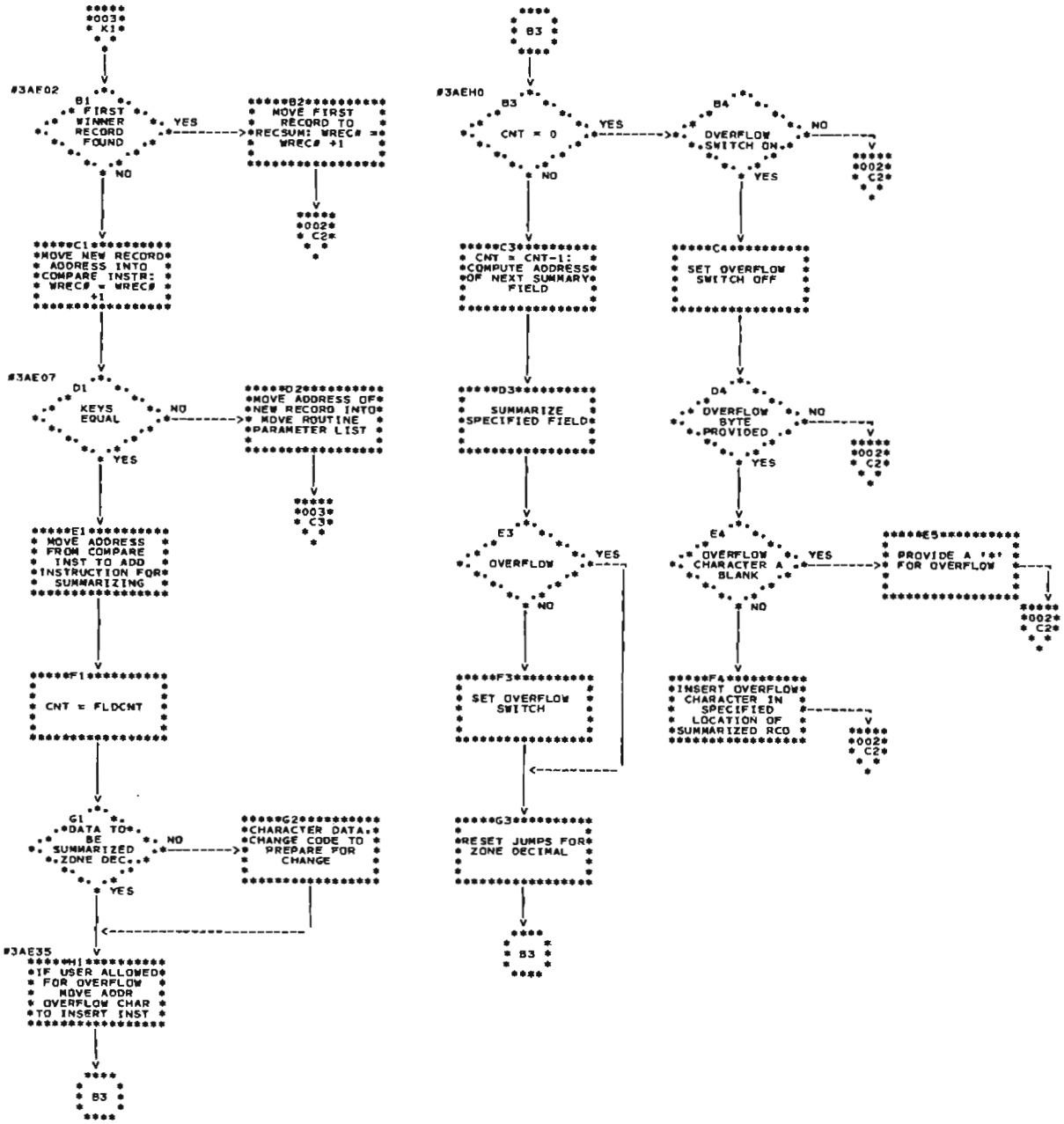
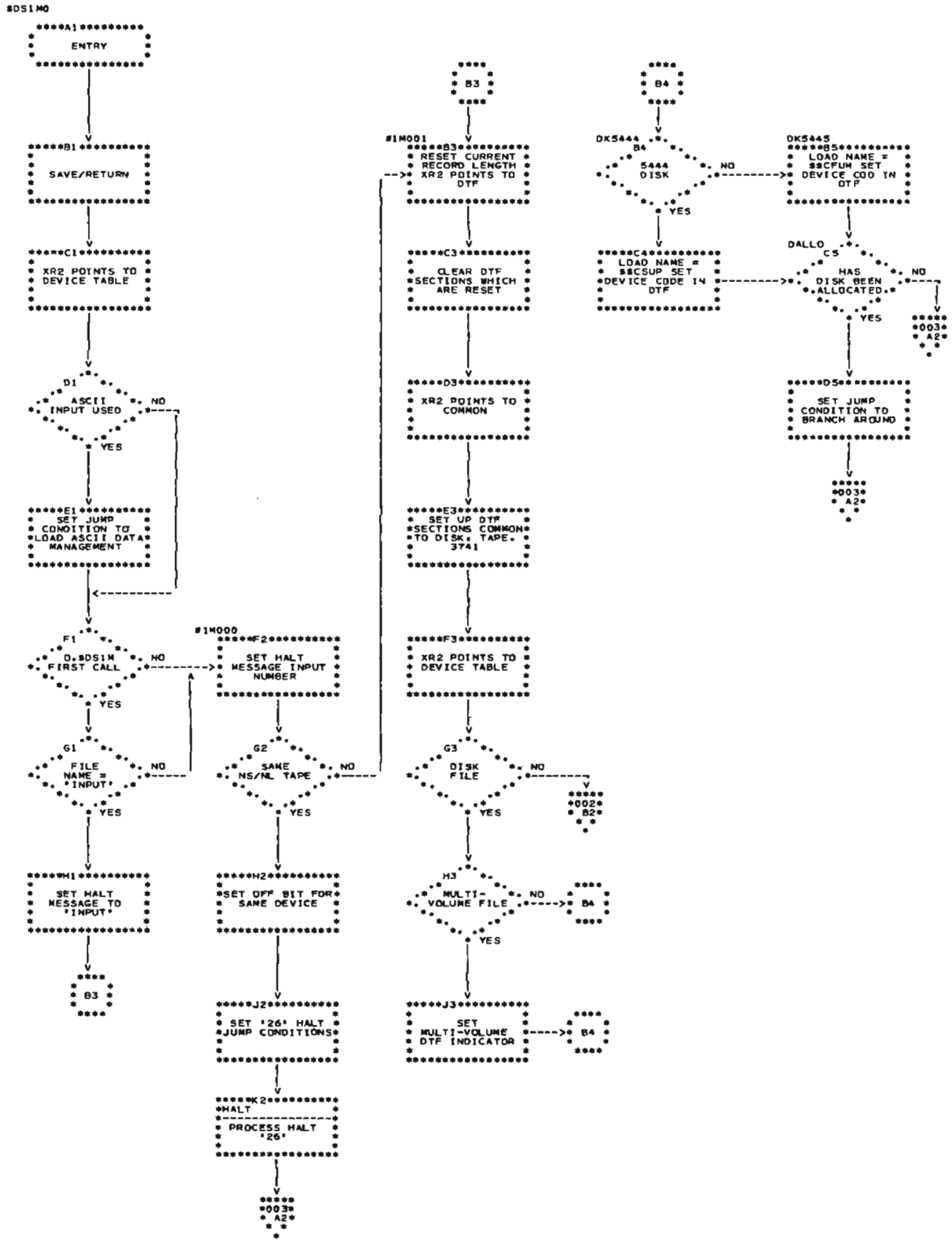
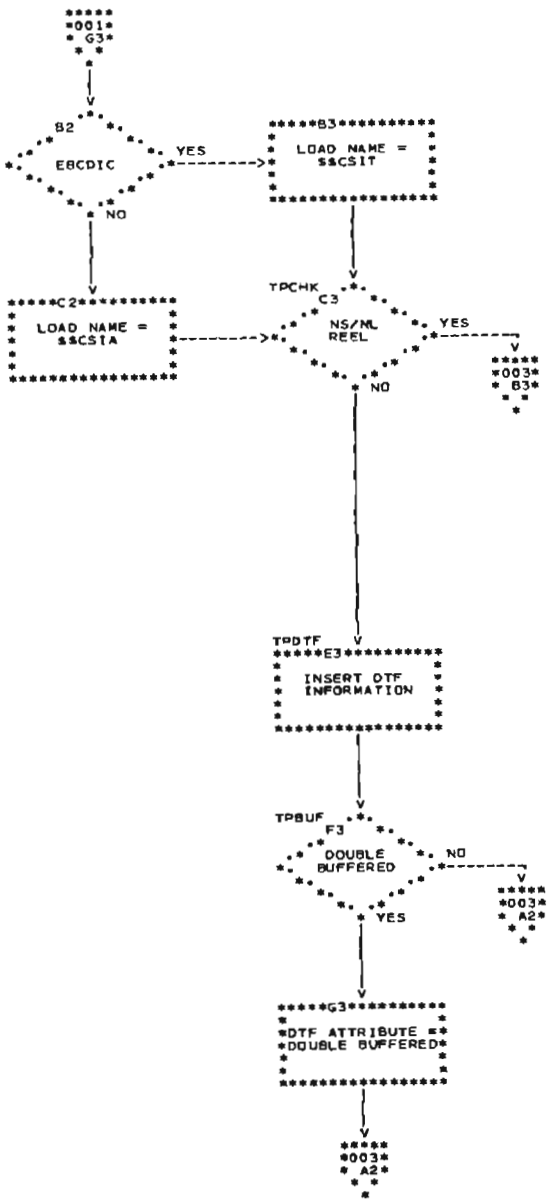


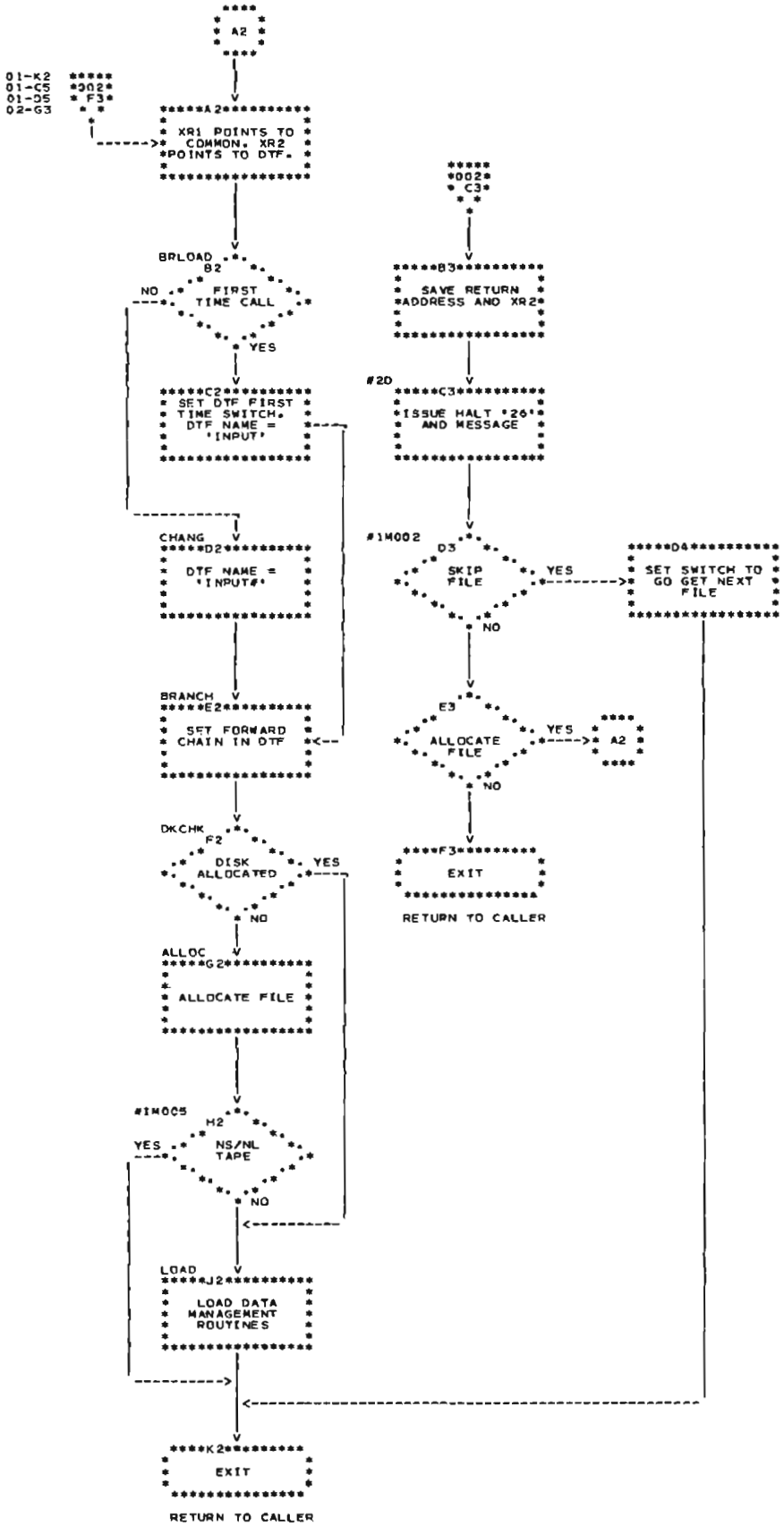
Chart CC (Part 4 of 4). Phase 3S – O.$DS3S – (All Models)



- Chart CD (Part 1 of 3). Phase 1M – O.$DS1M – (Model 12)



- Chart CD (Part 2 of 3). Phase 1M -- O.$DS1M -- (Model 12)



- Chart CD (Part 3 of 3). Phase 1M – O.$DS1M – (Model 12)

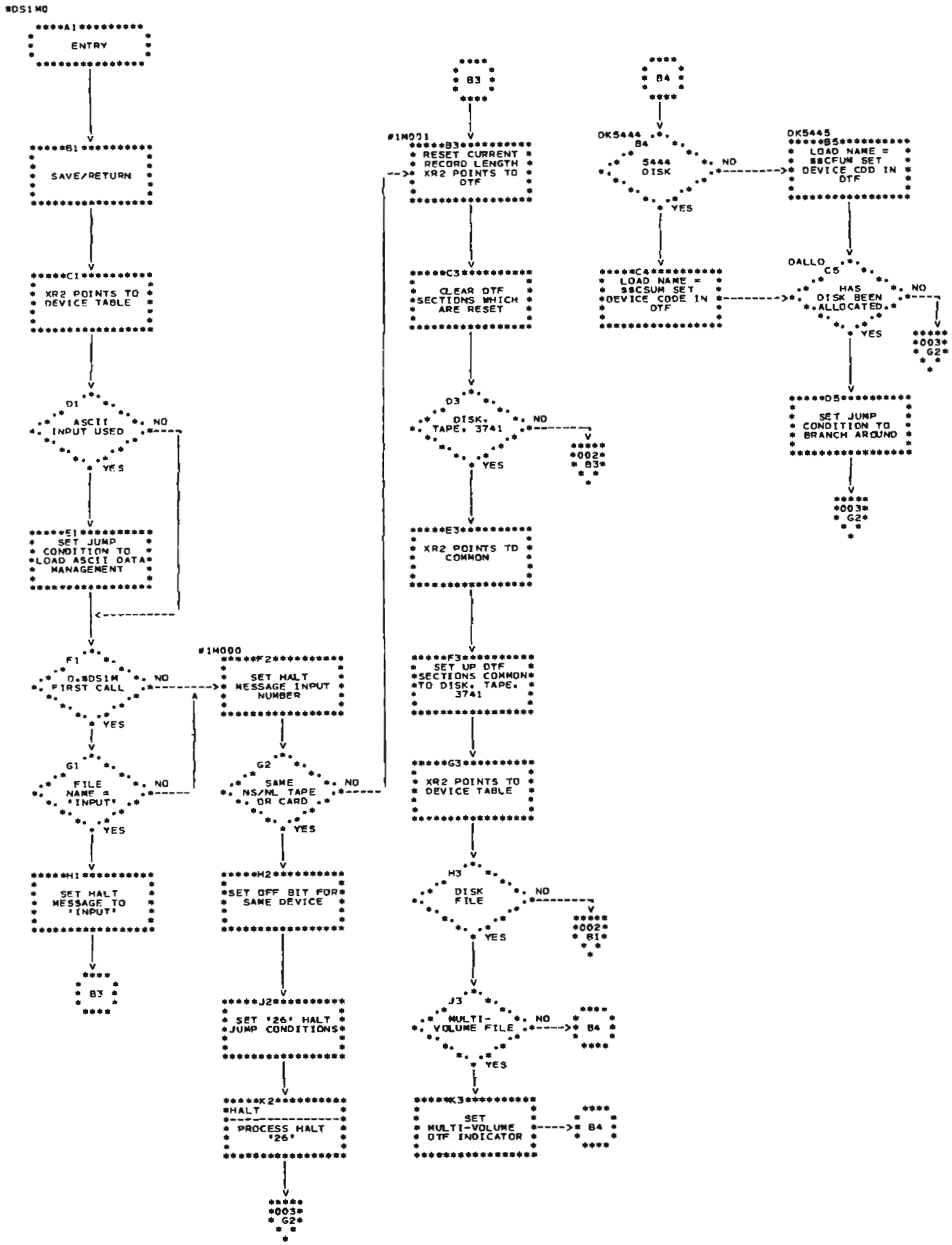


Chart CE (Part 1 of 3). Phase 1M – O.$DS1M – (Model 15)

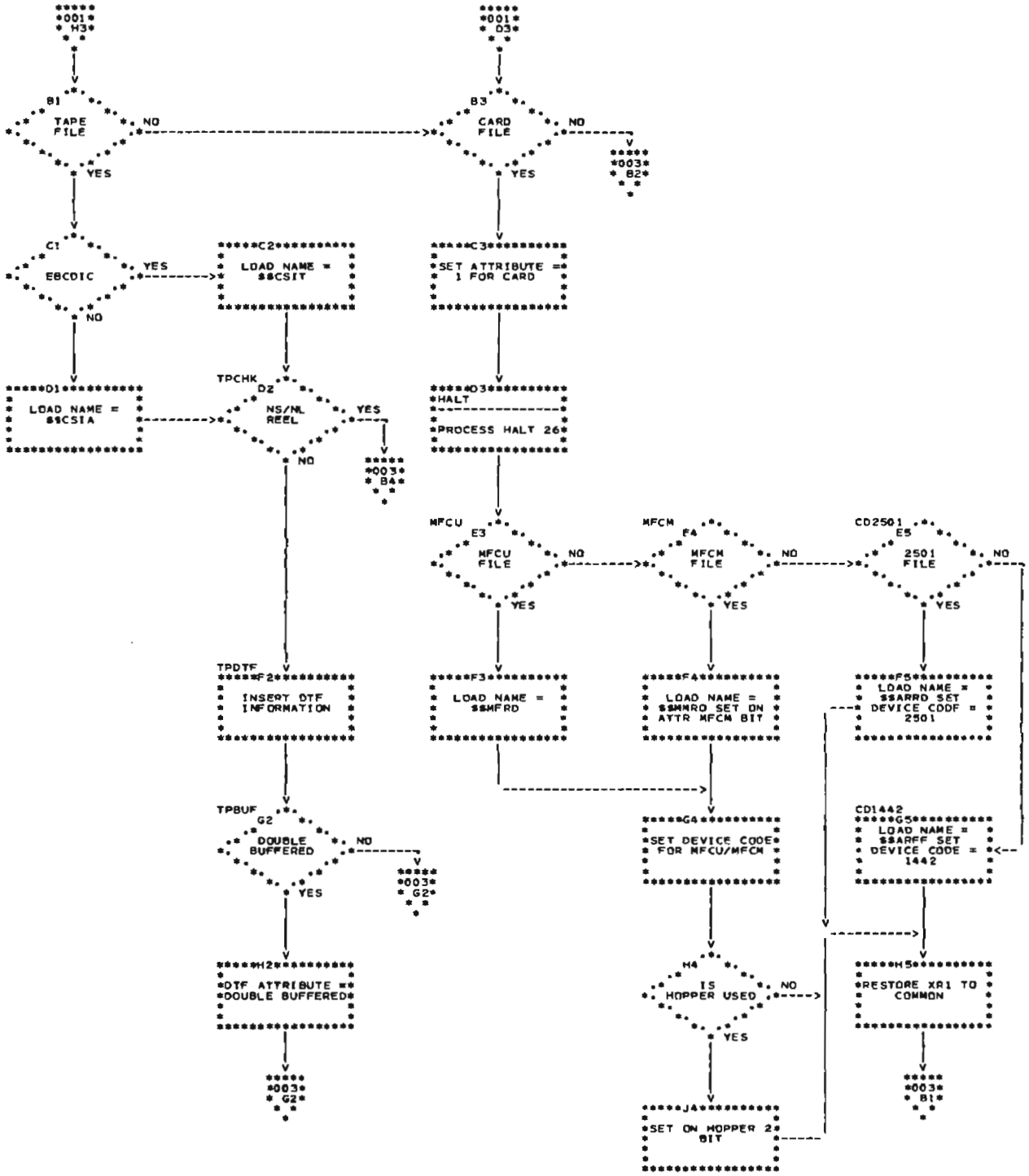


Chart CE (Part 2 of 3). Phase 1M – O.$DS1M – (Model 15)

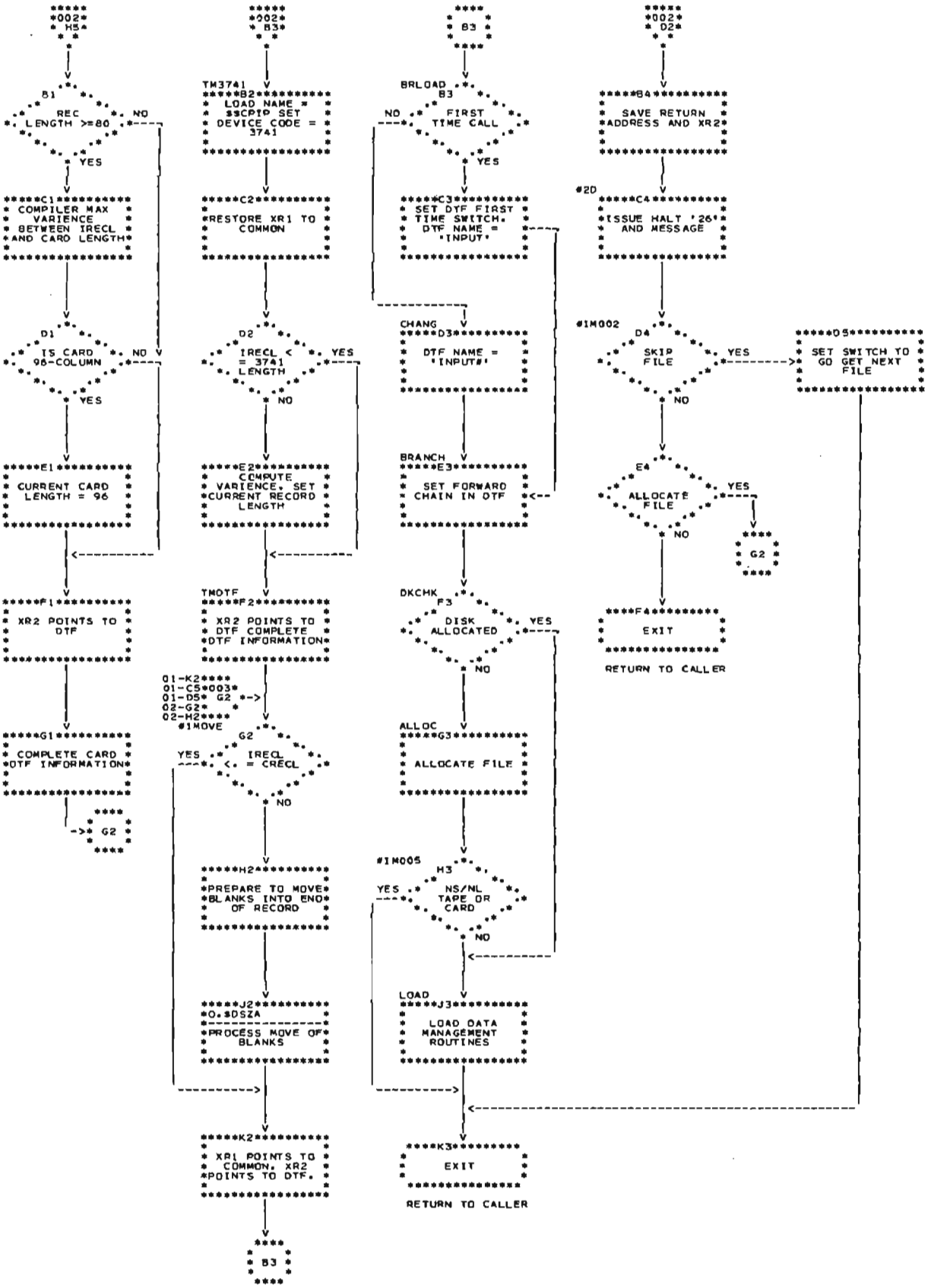


Chart CE (Part 3 of 3). Phase 1M – O.$DS1M – (Model 15)

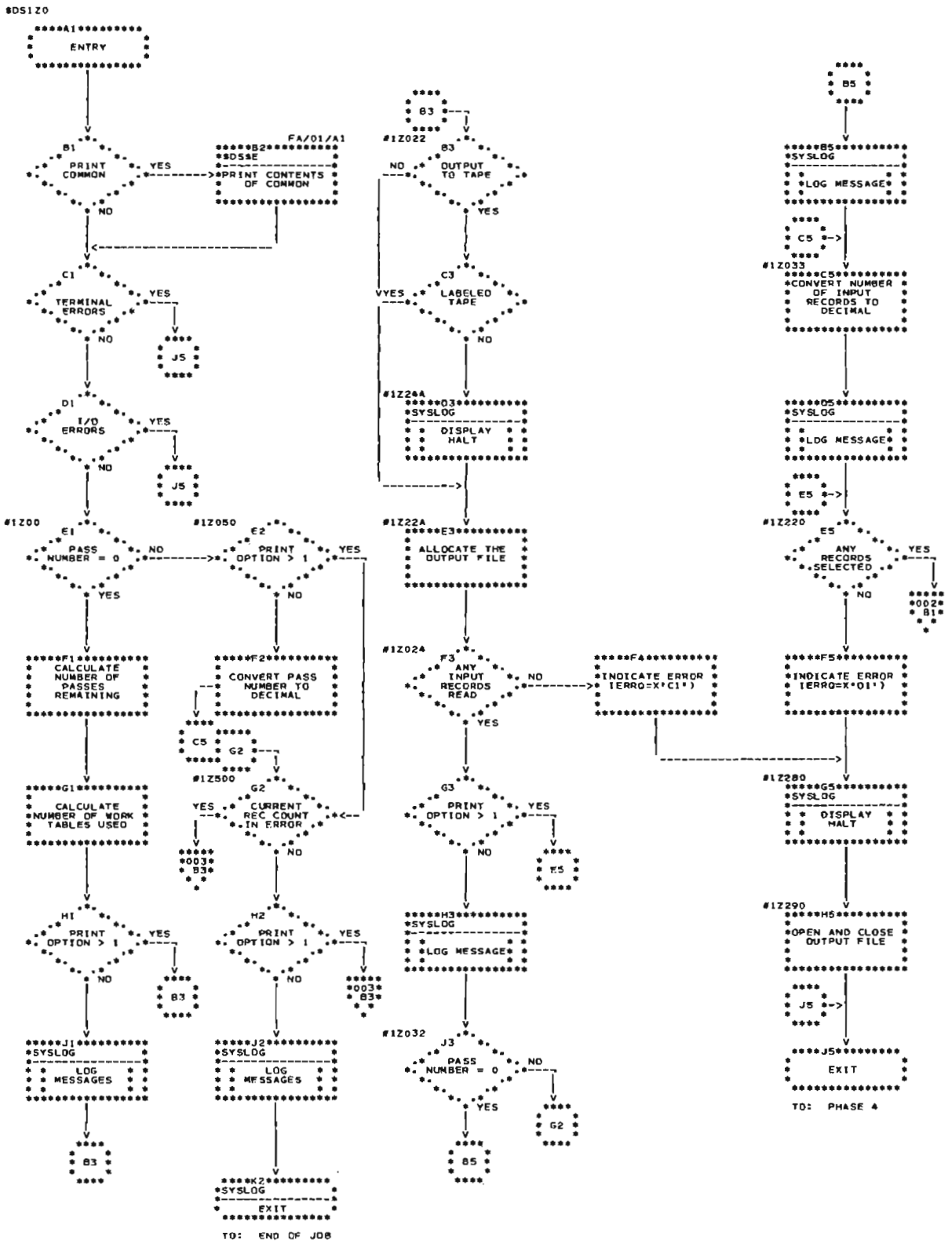


Chart DA (Part 1 of 3). End-of-Pass Reporter – O.$DS1Z – (Model 6, Model 10 Disk System)

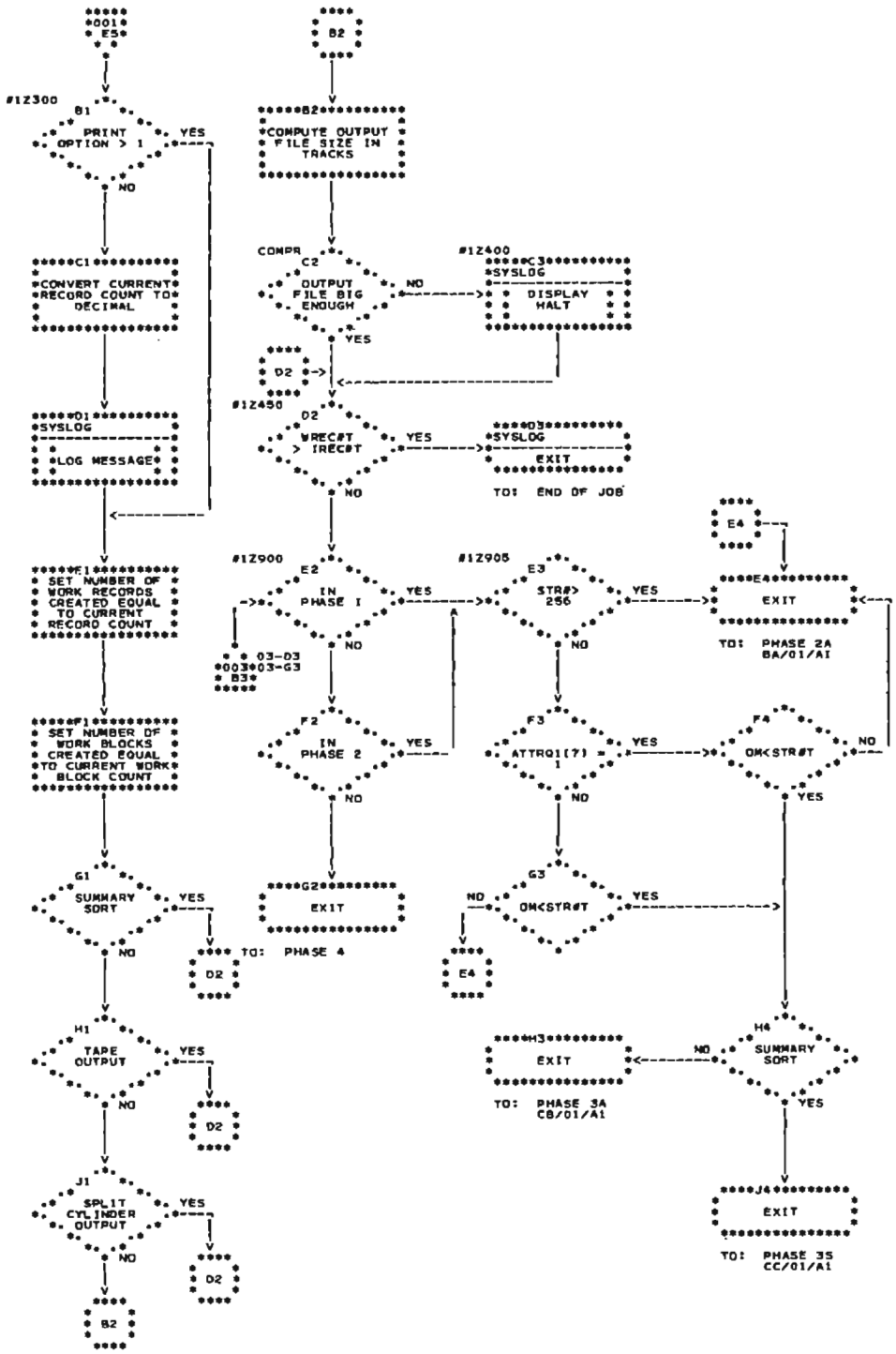


Chart DA (Part 2 of 3). End-of-Pass Reporter – O.$DS1Z – (Model 6, Model 10 Disk System)

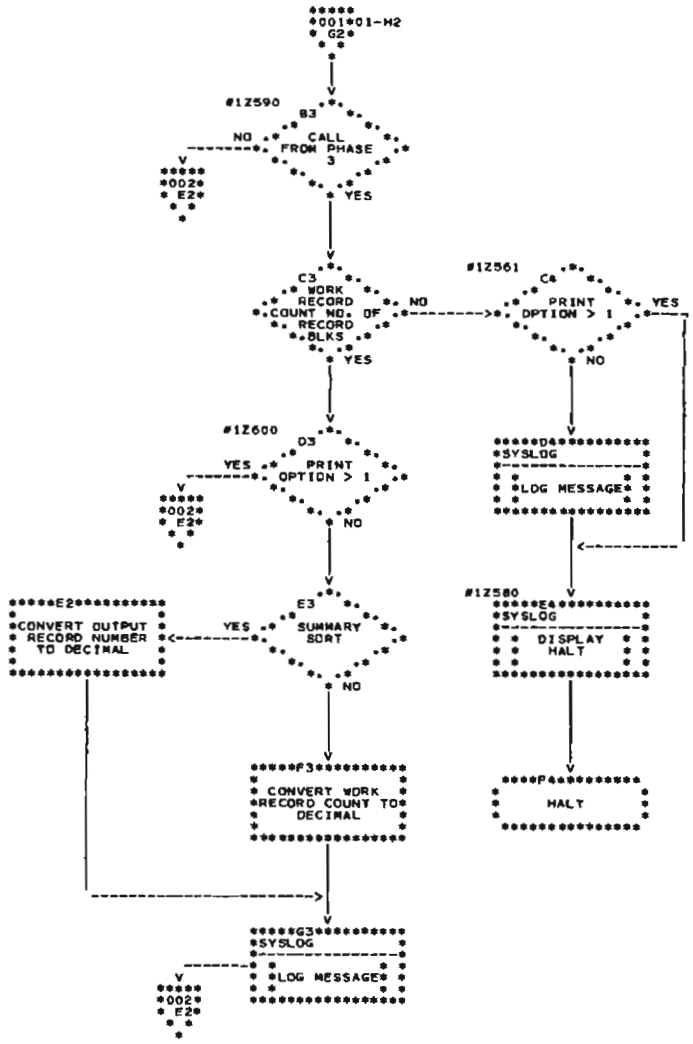


Chart DA (Part 3 of 3). End-of-Pass Reporter – O.$DS1Z – (Model 6, Model 10 Disk System)

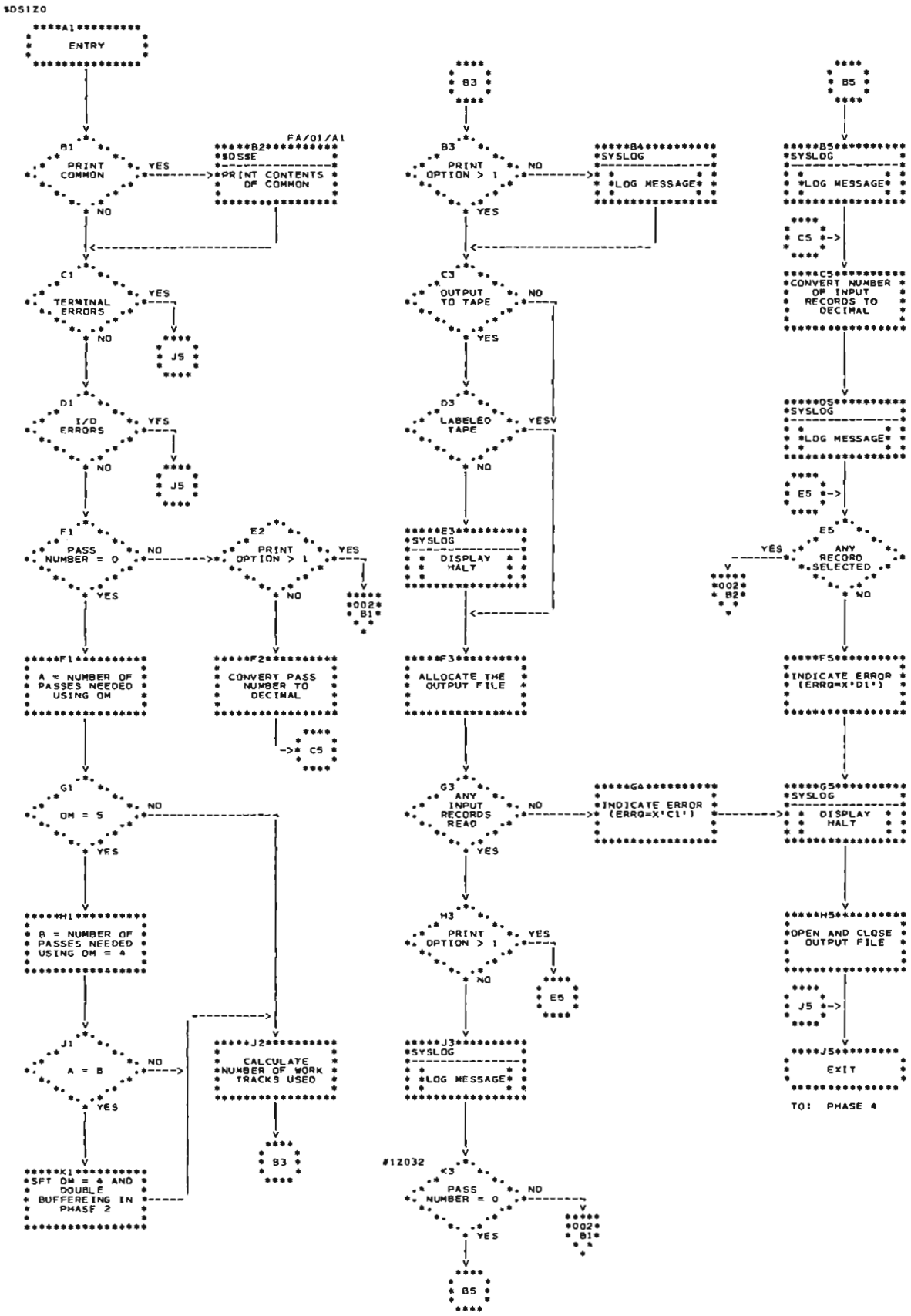


Chart DB (Part 1 of 3). End-of-Pass Reporter – O.$DS1Z – (Model 12, Model 15)

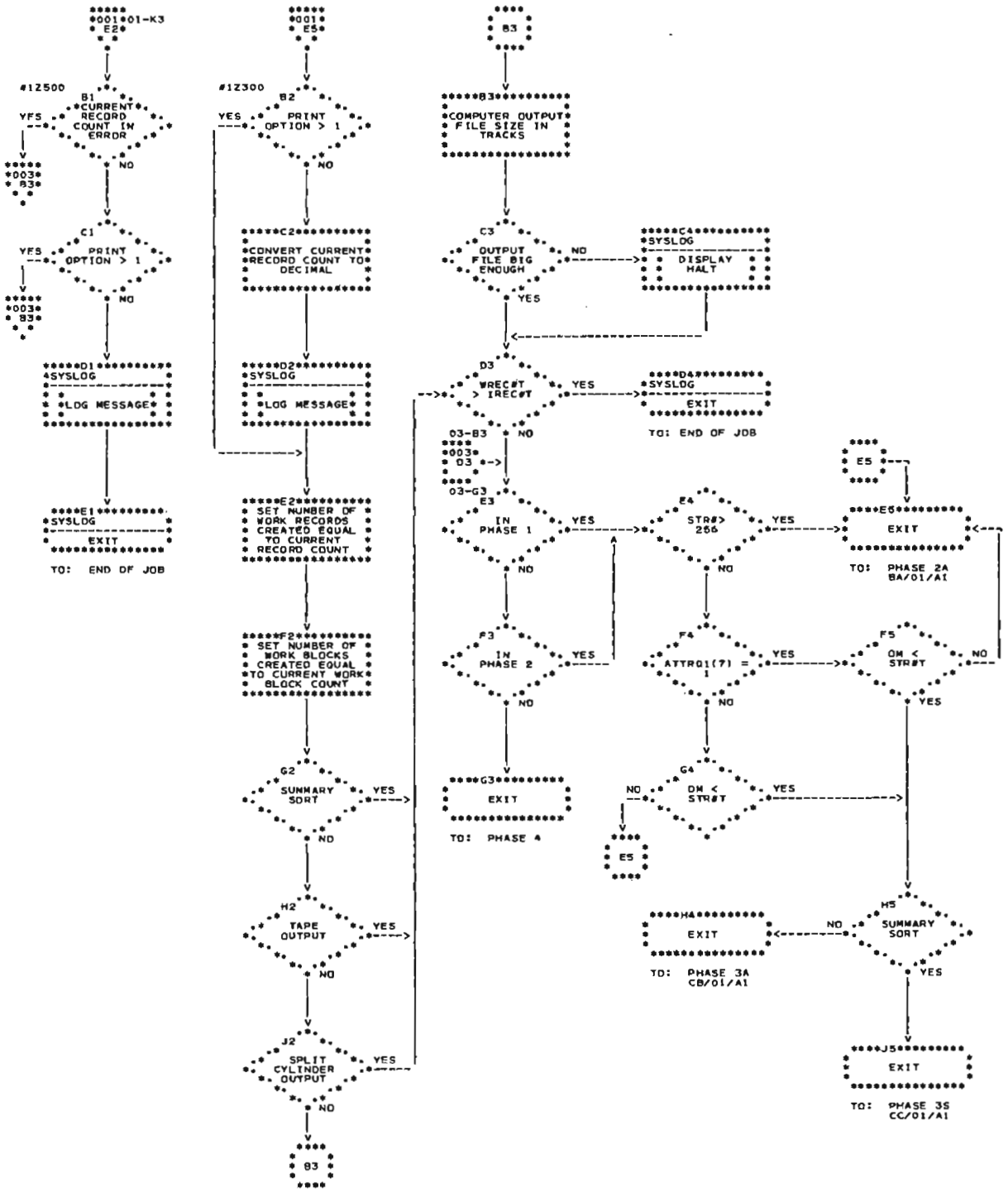


Chart DB (Part 2 of 3). End-of-Pass Reporter – O.$DS1Z – (Model 12, Model 15)

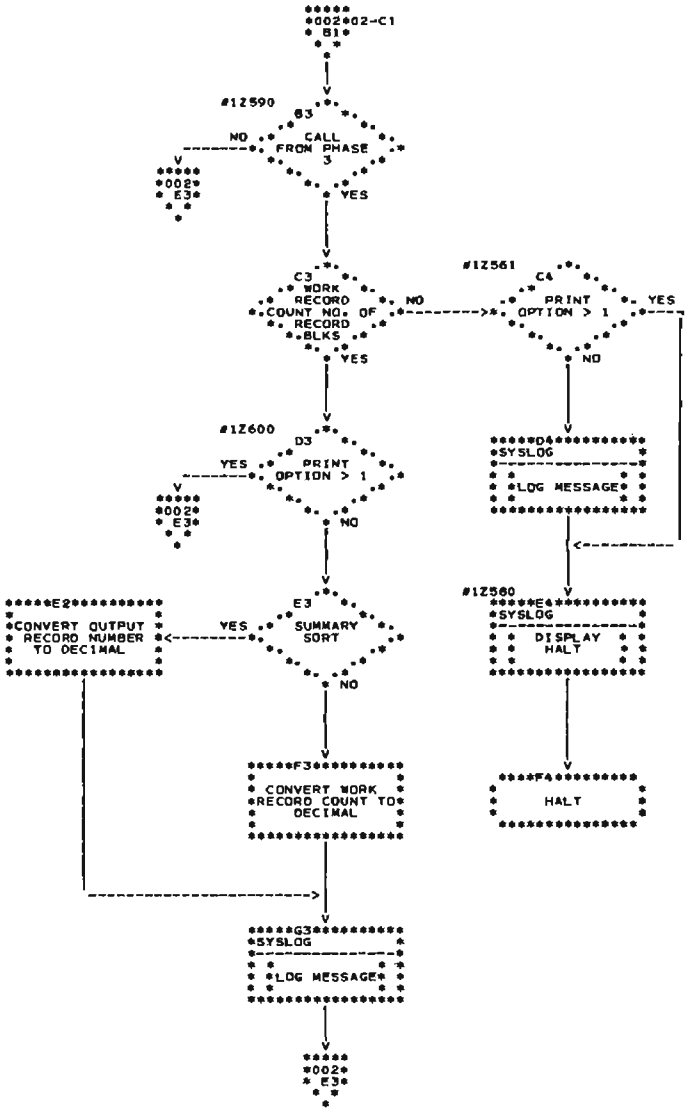


Chart DB (Part 3 of 3). End-of-Pass Reporter – O.$DS1Z – (Model 12, Model 15)

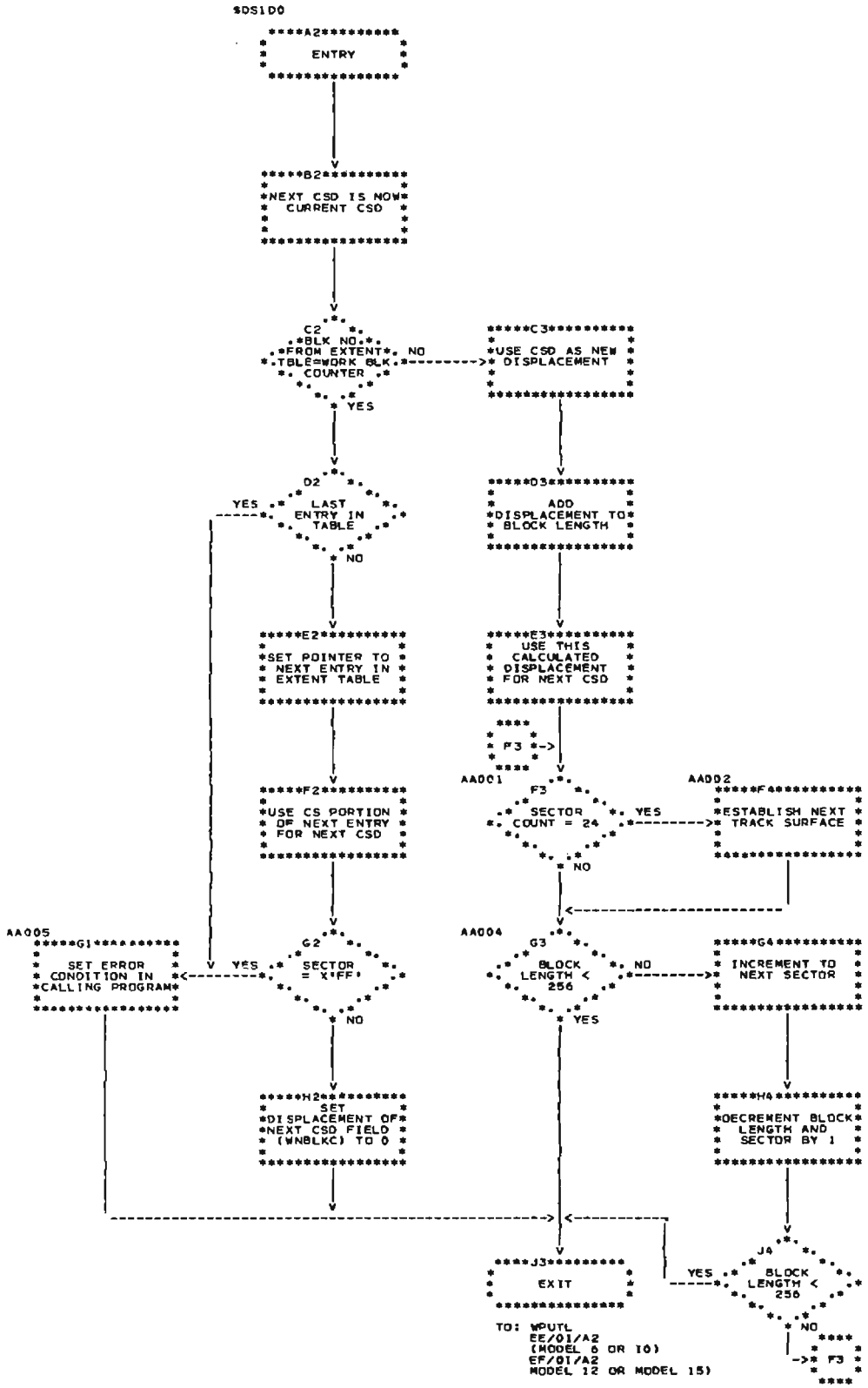


Chart EA. NEXTDB Routine for Phase 1 – O.$DS1D – (All Models)

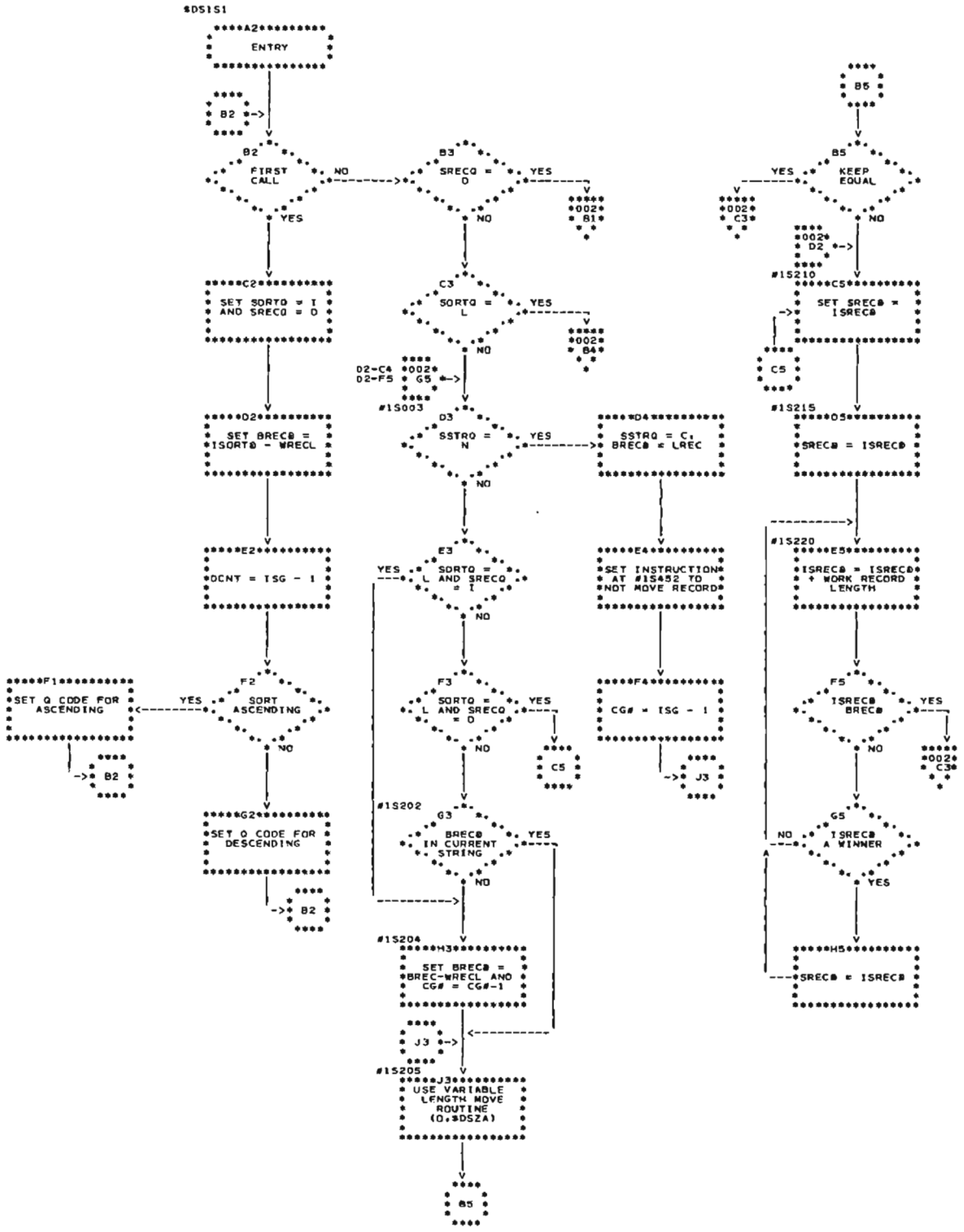


Chart EB (Part 1 of 2). Replacement Selection Internal Sort Routine – O.$DS1S – (All Models)

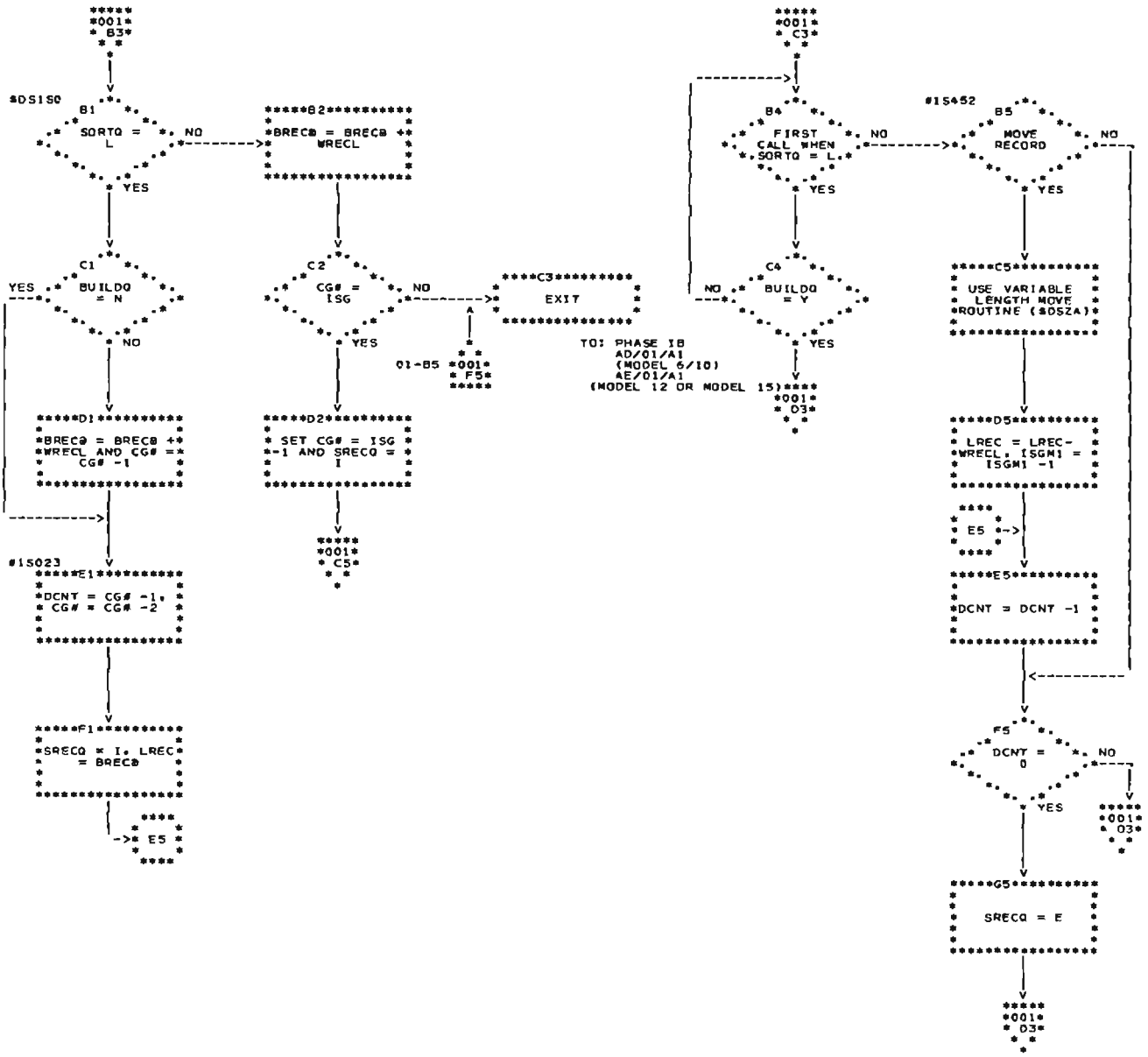


Chart EB (Part 2 of 2). Replacement Selection Internal Sort Routine – O.$DS1S – (All Models)

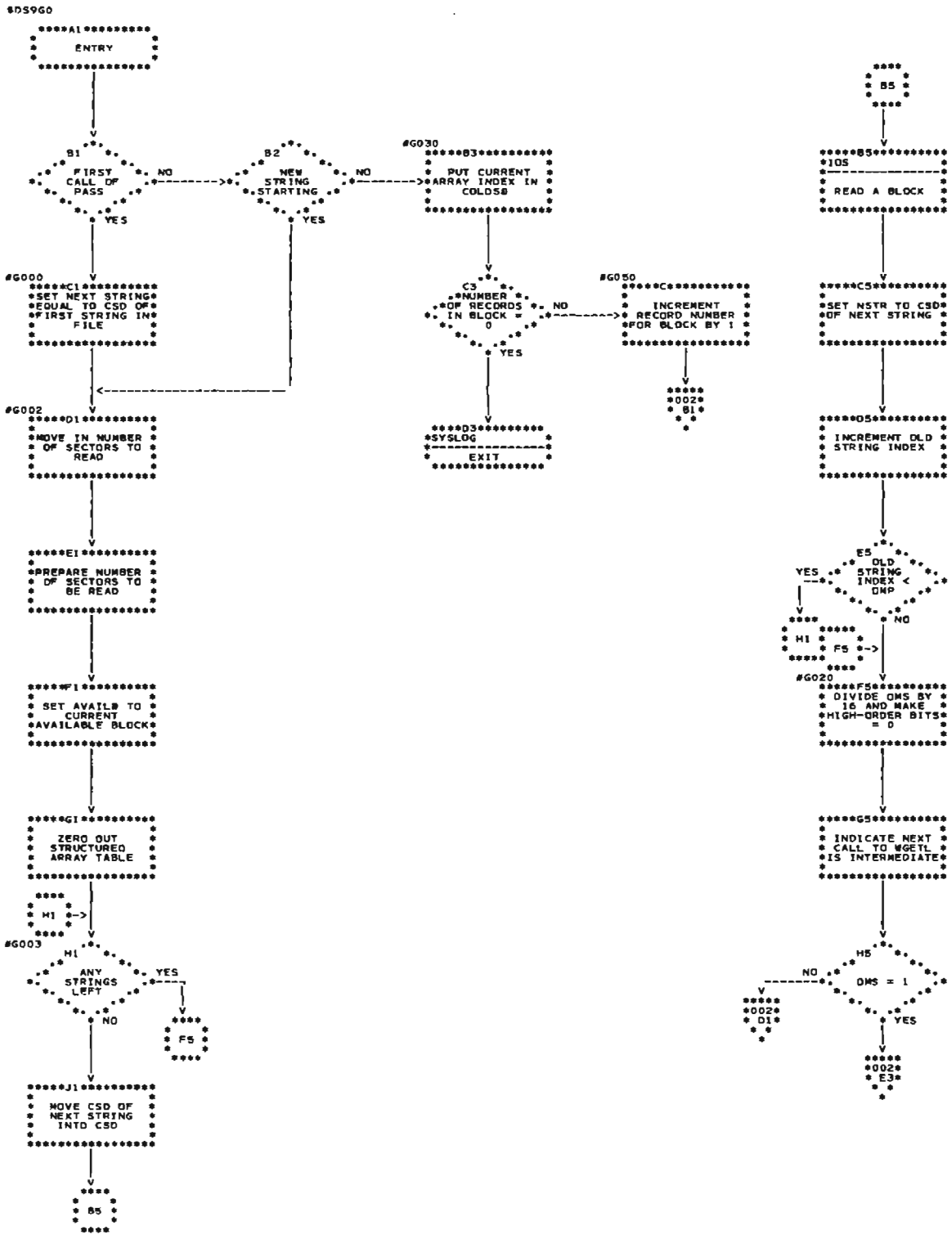


Chart ED (Part 1 of 2). WGETL Routine – O.$DS9G – (All Models)

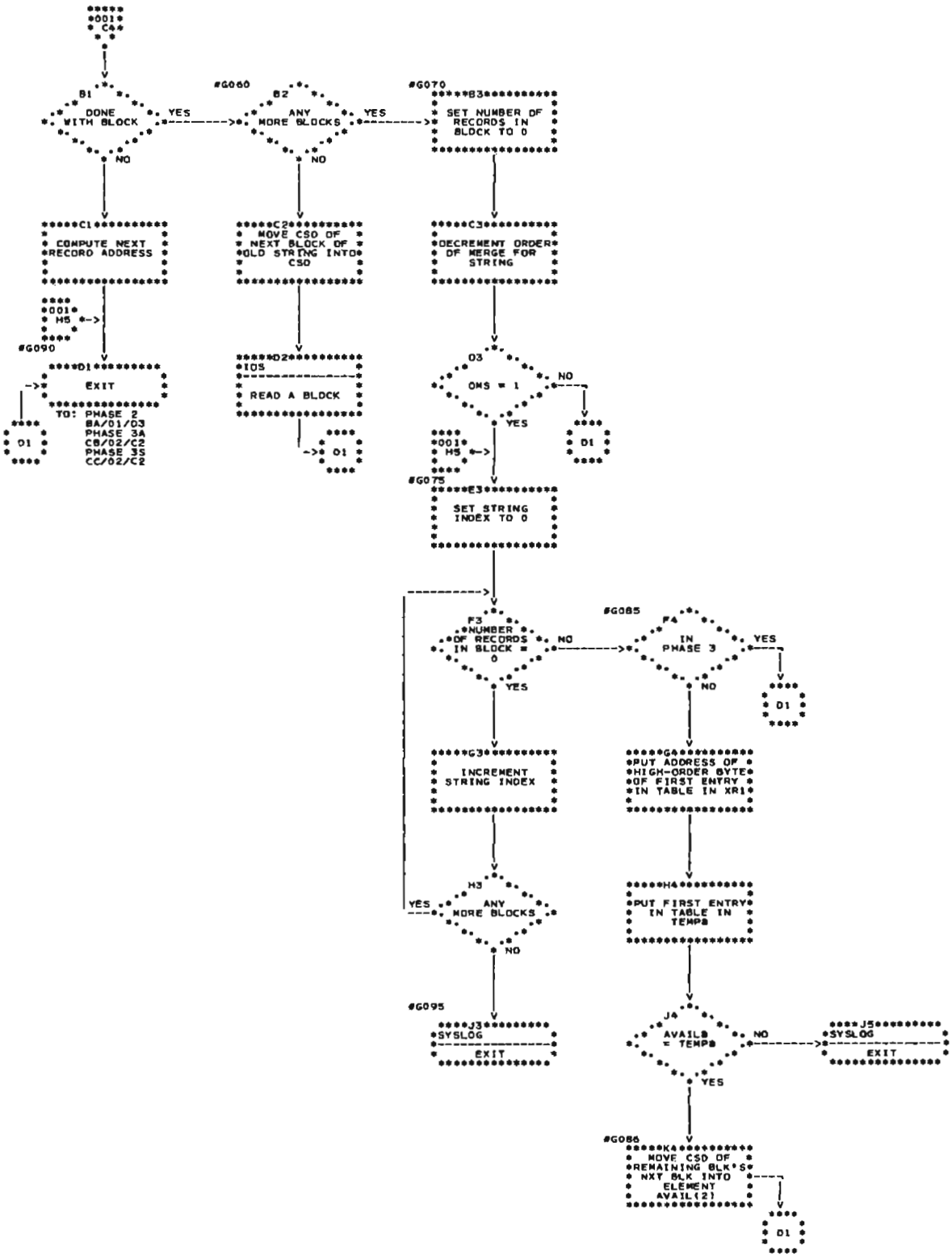


Chart ED (Part 2 of 2). WGETL Routine – O.$DS9G – (All Models)

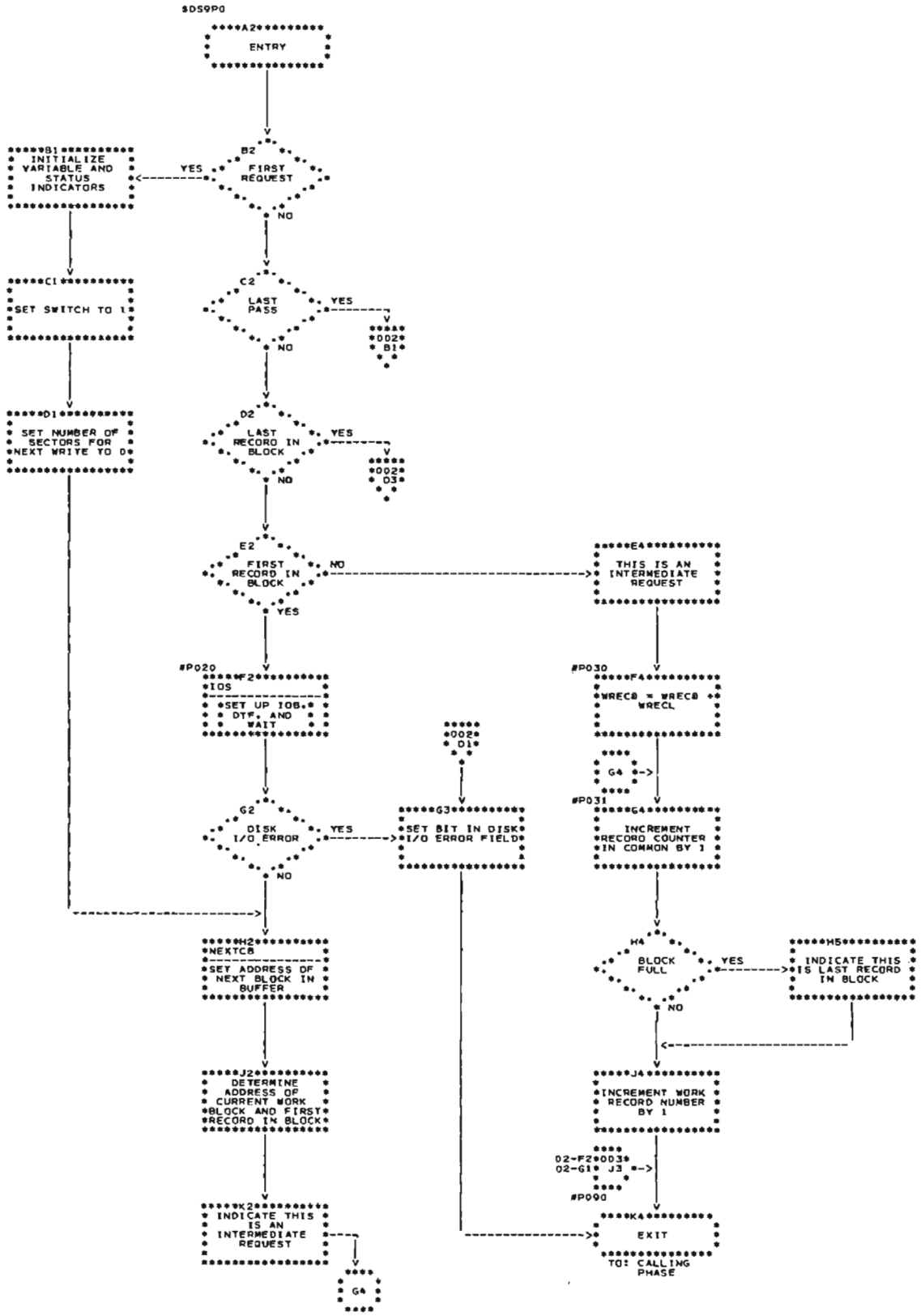


Chart EE (Part 1 of 3). WPUTL Routine – O.$DS9P – (Model 6, Model 10 Disk System)

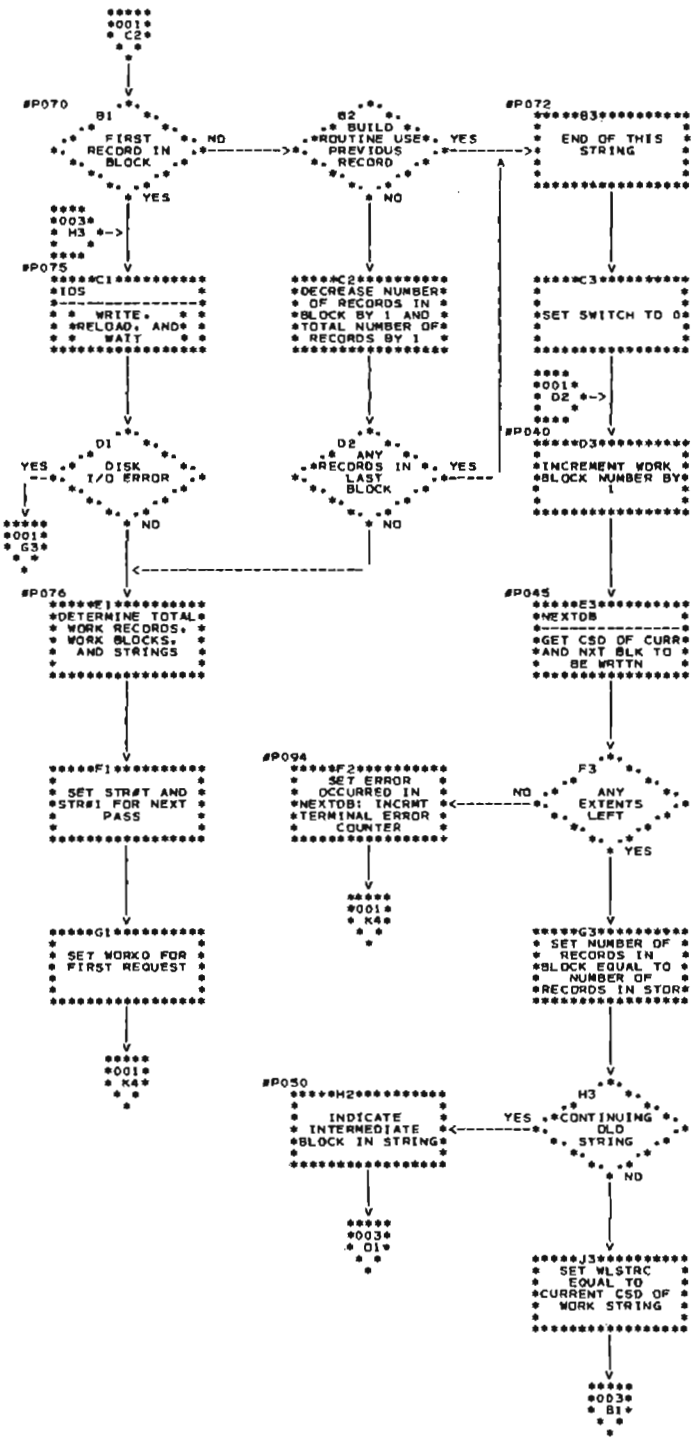


Chart EE (Part 2 of 3). WPUTL Routine – O.$DS9P – (Model 6, Model 10 Disk System)

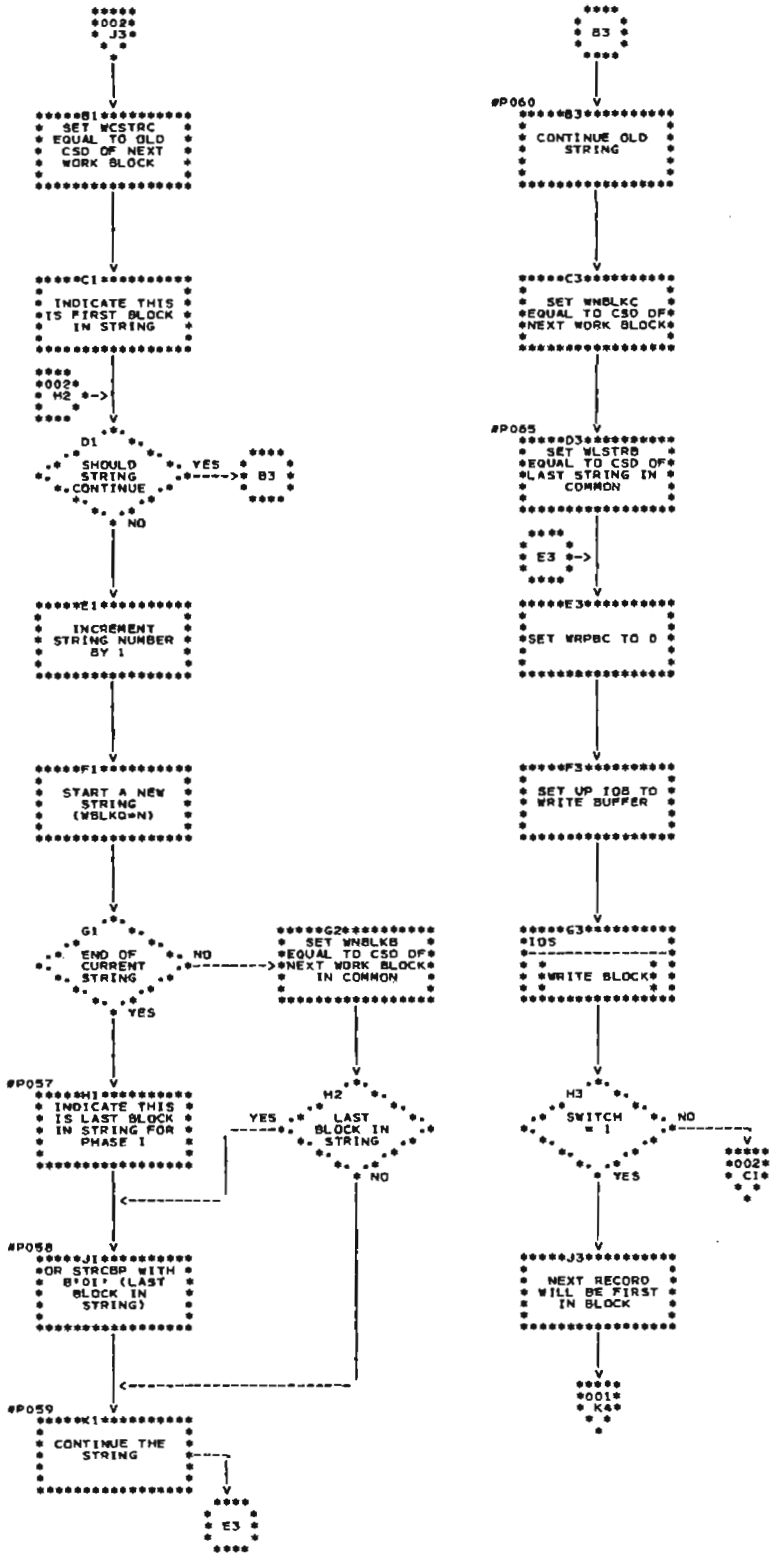


Chart EE (Part 3 of 3). WPUTL Routine – O.$DS9P – (Model 6, Model 10 Disk System)

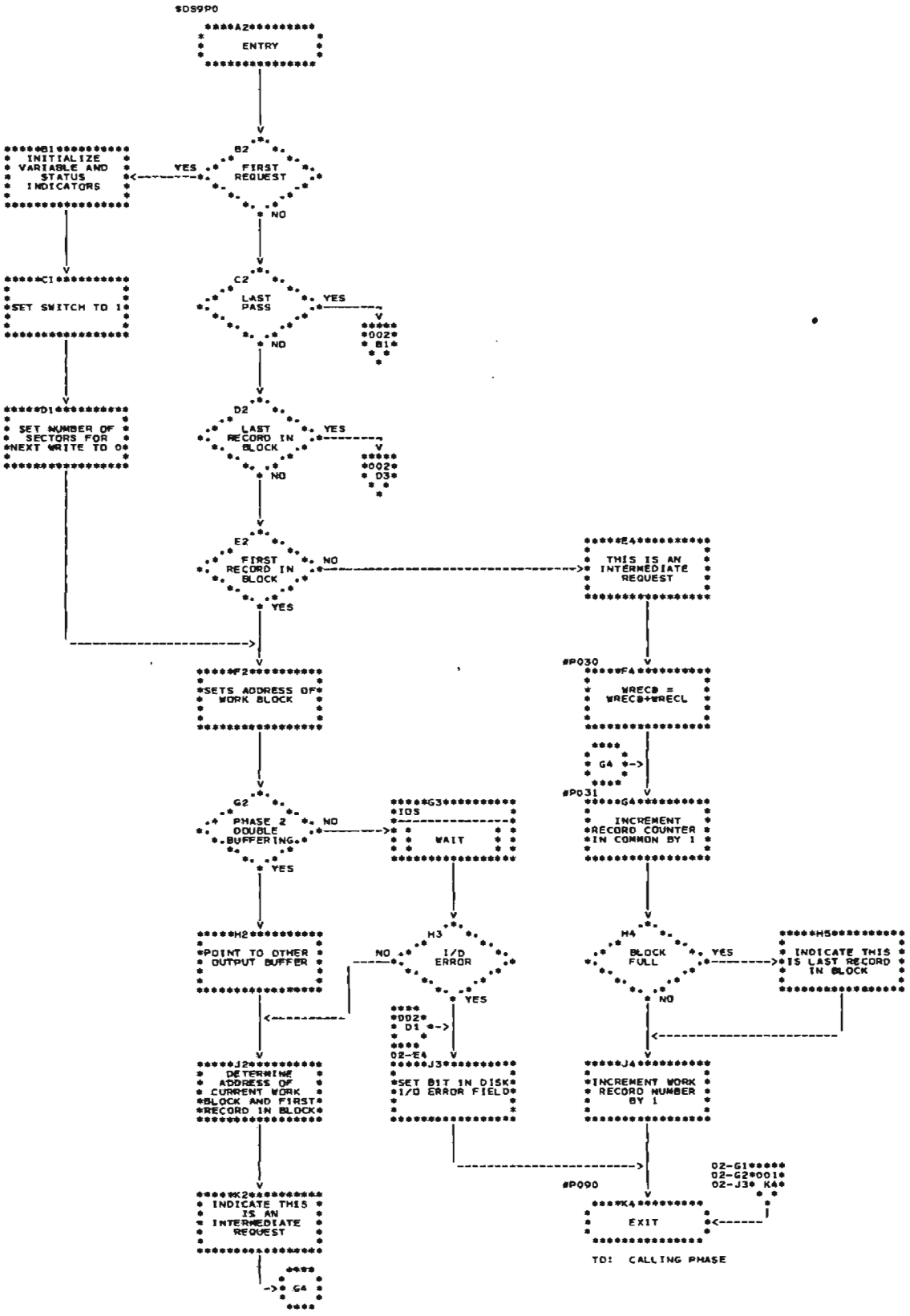


Chart EF (Part 1 of 3). WPUTL Routine – O.$DS9P – (Model 12, Model 15)

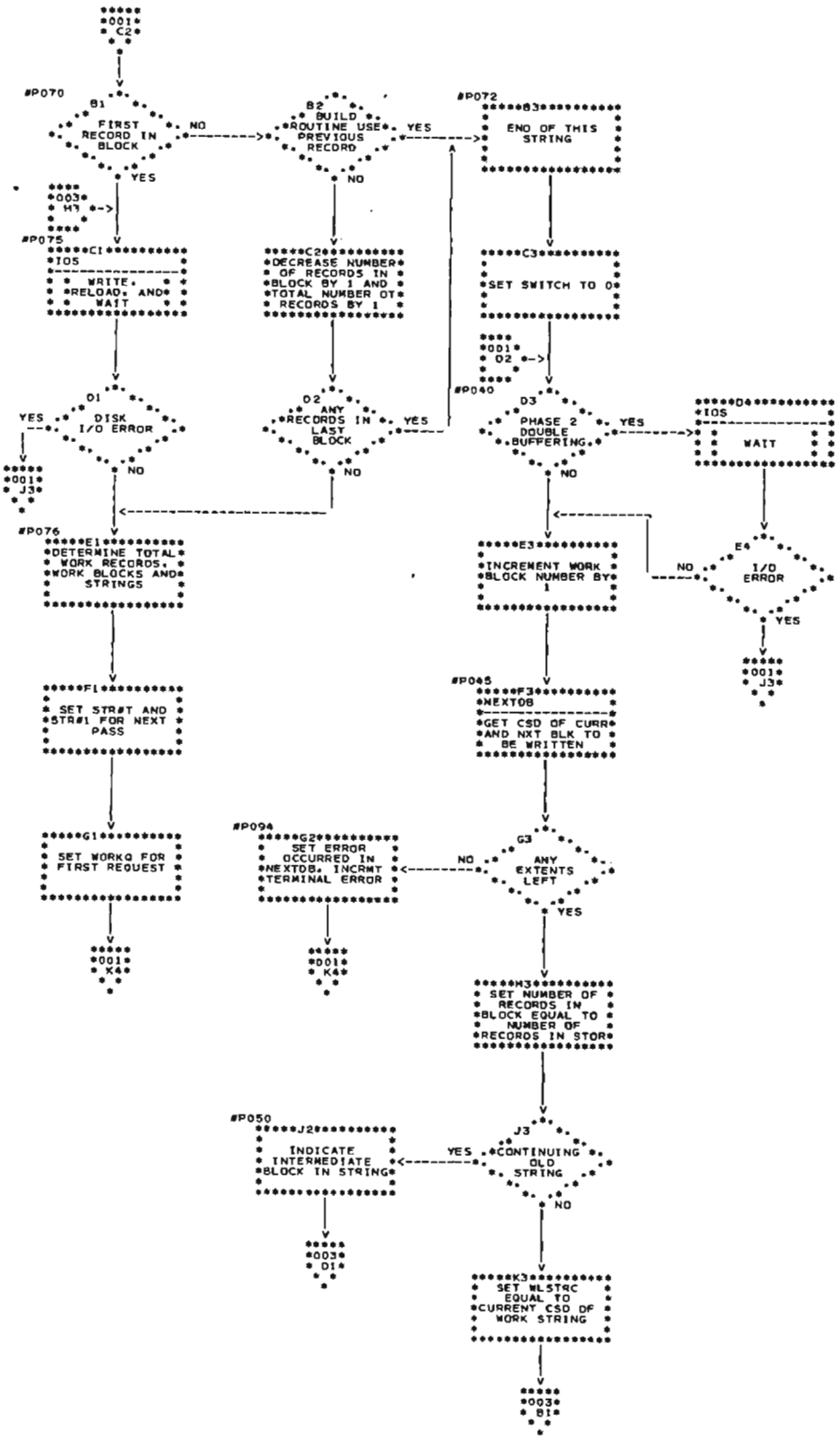


Chart EF (Part 2 of 3). WPUTL Routine -- O.$DS9P – (Model 12, Model 15)

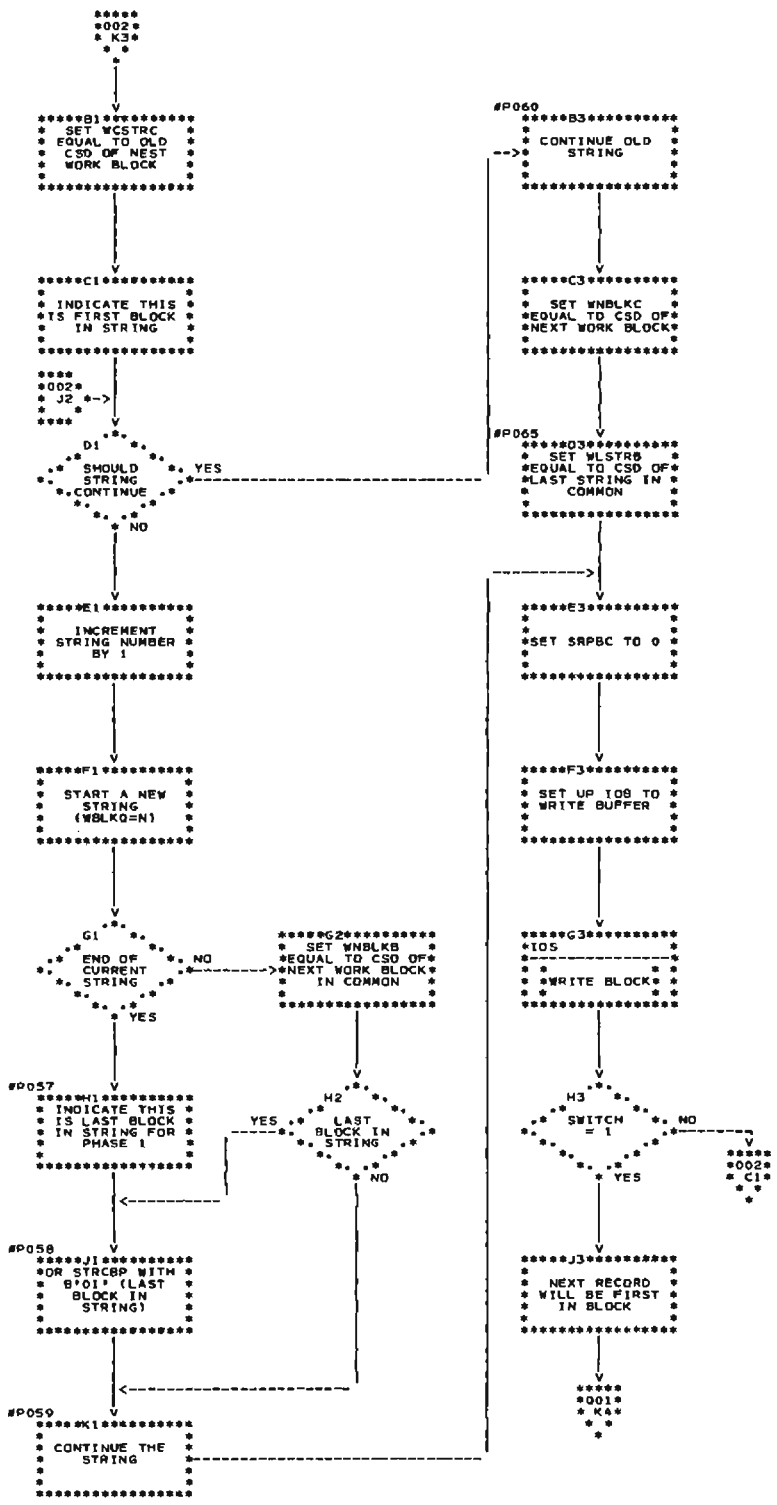


Chart EF (Part 3 of 3). WPUTL Routine – O.$DS9P – (Model 12, Model 15)

For quick reference to the program listings on microfiche, the directory lists the phases and routines used in Disk Sort. The phases and routines are listed in alphameric order. The directory contains the following columns:

- *Name* is the symbolic label used to identify a phase or routine. This name appears at the beginning of the prologue for that module in the microfiche listing.
- *Entry Point* is the symbolic label of the first executed instruction in the phase or routine. This is the point at which the phase or routine is entered from another phase or routine.
- *Descriptive Name* is the specific name used in this publication to better identify the phase or routine.
- *Synopsis* is a brief summary of the main functions performed by the phase or routine.

Generation Phases

Name	*Entry Point*	*Descriptive Name*	*Synopsis*
$DSORT		Phase ØA	Made up of routines $DSAA, $DSAB, and $DSAF.
$DSAA	$DSAAØ	Mainline routine	Initializes part of COMMON; reads first sequence specification statement (header).
$DSAB	$DSABØ	Locate Control Statements routine	Locates the sort specification statements and loads in O.$DS9I, O.$DS9W, or O.$DS9S to read them.
$DSAF	$DSAFØ	Check for Work File Statement routine	Determines whether there is a work file statement; determines if automatic work file allocation is being used; determines if input and output files are online.
			Builds the Device Table (Model 12 and Model 15).
$DSBA	$DSBAØ	Phase ØB	Initializes rest of COMMON; determines type of sort; determines if sort is ascending or descending; loads in O.$DSBC if alternate collating sequence requested.
			Reformats the Device Table (Model 12 and Model 15).
$DSCA		Phase ØC—Part A (Compile the Select/Build routine)	Made up of the routines $DSCA1 through $DSCA9.
$DSCA1	$DSCAØ	Initialization routine	Initializes Phase ØC COMMON fields; checks for and prints all bypassed comments or data statements to get first include, omit, or field statement.

Name	*Entry Point*	*Descriptive Name*	*Synopsis*
$DSCA2	$DSCA2	Mainline routine	Reads and prints specification statements: determines if statement is include, omit, or field statement, and checks validity of the statement.
$DSCA5	$DSCA5	Code Segment Move routine	Moves generated code to the temporary Select/Build area in storage.
$DSCA6	$DSCA6	Determine Zone routine	Determines zone of a given byte.
$DSCA8	$DSCA8	Error Table Build routine	Builds the error table for use by O.$DSCB.
$DSCE	$DSCE	Include/Omit Generator routine	Generates proper code segments in the Select/Build routine for include and omit statements.
$DSCF	$DSCF	Field Generator routine	Generates proper code segments in the Select/Build routine for field statements; generates summary table.
$DSCL	$DSC9Ø	Compute specified lengths	Made up of routines $DSCA7 and $DSCA9.
$DSCA9	$DSCA9	Calculate Length routine	Calculates the length and displacement of Factor 1, Factor 2, and control word area.
$DSCA7	$DSC7Ø	Decimal to Binary Convert routine	Converts the To field of Factor 1 to its binary value.
$DSCZ	$DSCZØ	End of File for Compiler routine	Finalizes Select/Build code; updates phase load address in COMMON; moves summary table to Phase 1 location behind COMMON; gives control to $DSCB immediately, if too many errors (CASW1 bit 7=1).
$DSCB	$DSCBØ	Phase ØC–Part B (Error Print)	Writes messages for errors found in Part A of Phase ØC.
$DSCC	$DSCCØ	Phase ØC–Part C (Move Generation Code)	Moves generated code for the Select/Build routine to its permanent storage location; moves fixed code in place; optionally moves Pack-Include/Omit and Pack-Field routines in place, and optionally moves the summary table behind the Select/Build routine. Moves the Device Table behind the summary table (Model 12 and Model 15).

Name	*Entry Point*	*Descriptive Name*	*Synopsis*
$DSDA	$DSDAØ	Phase ØD--Part A	Determines active program lengths of the execution phases.
$DSDB	$DSDBØ	Phase ØD--Part B	Determines type of sort and order of merge to be used; determines the length of the input buffer, work buffer, work block, internal sort area, output buffer, and output record.
$DSEA	$DSEAØ	Phase ØE	Allocates output file if not a deferred mount; calls automatic allocate ($DSEG) if work file statement is omitted; converts sectors per extent in table of work file extents to blocks per extent.
$DSGA	$DSGAØ	Phase ØG	Gives control to proper execution phase; writes descriptive error and information messages.

Name	*Entry Point(s)*	*Descriptive Name*	*Synopsis*
$DSBC	$DSBCØ	Alternate Collating Sequence Table routine	Contains the alternate collating sequence table; modifies the table.
	$DSBC1		Uses the table to translate data.
$DSEG	$DSEGØ	Automatic Work File Allocation routine	Allocates work file space; builds table of work file extents.
$DSZB	$DSZBØ	Decimal to Hex Conversion routine	Converts a 7-byte zoned decimal number to a 3-byte hex number.
$DSZC	$DSZCØ	Hex to Decimal Conversion routine	Converts a 3-byte hex number to a 7-byte zoned decimal number.
$DSZD	$DSZDØ	Four-Byte Hex Divide routine	Divides one 4-byte hex number by another.
$DSZM	$DSZMØ	Three-Byte Hex Multiply routine	Multiplies two 3-byte hex numbers.
$DS9I	$DS9IØ	Read Statement from SYSIN Reader routine	Reads statements from the system input device; indicates if the read was successful and if end of file has occurred.
$DS9W	$DS9WØ	Read Statement from Scheduler Work Area routine	Reads statement images located in the scheduler work area; indicates if the read was successful and if end of file has occurred.
$DS9S	$DS9SØ	Read Statement from Source Library routine	Reads next specifications statement by branching to the Source Library Get routine; checks the return code from the Source Library Get routine.

Execution Phases

Name	*Entry Point*	*Descriptive Name*	*Synopsis*
$DS1L	$DS1LØ	Phase 1L	Loads modules required for Phase I into main storage; performs initialization common to O.$DS1A, O.$DS1B, and O.$DS1X; passes control to the required sort algorithm.
$DS1A	$DS1AØ	Phase 1A	Performs an internal sort external to the work block when there is not enough storage to create strings of sequenced work records on the work file, each string one block in length.
$DS1B	$DS1BØ	Phase 1B	Performs an internal sort external to the work block when there is enough storage to create variable length strings of sequenced work records on the work file.
$DS1S	$DS1SØ	Replacement Selection Internal Sort routine	Performs a selection internal sort for Phase 1B.
$DS1X	$DS1XØ	Phase 1X	Performs a tournament sort outside of the work block if it is determined that a tournament sort will run faster than the sort technique used in Phase 1A or 1B; produces variable length strings of sequenced work records.
$SD1M	$DS1M0 (Model 12 and Model 15)	Next file initiator	Sets up the DTF, allocates the file, and loads Data Management for the next file to be processed as input.
$DS1Z	$DS1ZØ	End-of-Pass Reporter	Provides (end-of-pass) information for execution phases; loads in successive phases.
$DS2L	$DS2LØ	Phase 2L	Loads modules required for Phase II into main storage, then passes control to O.$DS2A.
$DS2A	$DS2AØ	Phase II	Merges strings on the work file of records created in Phase 1A, 1B, or 1X until the total number of strings is less than or equal to the sort's order of merge.
$DS3L	$DS3LØ	Phase III	Loads modules required for last pass into main storage; recomputes output buffer sizes; performs intialization common to O.$DS3A and O.$DS3S; then gives control to desired module.
$DS3A	$DS3AØ	Phase 3A	Performs the final merge for this sort; places sorted records consecutively on the output file.

Execution Routines

Name	*Entry Point*	*Descriptive Name*	*Synopsis*
$DS3S	$DS3SØ	Phase 3S	Performs the final merge for a summary sort, summarizing as the last pass is executing; places the summarized sorted records on the output file.
$DS4A	$DS4AØ	Phase IV	Writes error messages; ends job for normal conclusion or for error conditions.
$DSZA	$DSZAØ	Variable Length Move routine	Moves a given number of bytes from one storage area to another.
$DS1D	$DS1DØ	NEXTDB routine for Phase I	Gets the cylinder/sector/displacement for Phase 1A, 1B, or 1X of the next available consecutive block in the work file to write a block.
$DS9P	$DS9PØ	WPUTL routine	Locates storage position of the next record in the work block; writes full blocks.
$DS9G	$DS9GØ	WGETL routine	Gets storage address of next record to be merged in the indicated old string.

Diagnostic Aid Phases

Name	*Entry Point*	*Descriptive Name*	*Synopsis*
DSE	DSEØ	Dynamic Dump Loader	Determines if there is enough available storage to load O.DSA.
DSA	DSAØ	Print COMMON– Dynamic Request	Dumps COMMON after each phase if DEBUG specified on header statement and enough storage available to do dump; contains COMMON entries used by all phases.
DSL (Model 6 and Model 10 Disk System only)	DSLØ	Print COMMON- Terminal Request	Dumps COMMON and terminates the job.

Diagnostic Aid Routines

Name	*Entry Point*	*Descriptive Name*	*Synopsis*
$DSZE	$DSZEØ	BITOHEX routine	Converts 4-bit groups into their equivalent equivalent 8-bit EBCDIC values.
$DSZF	$DSZFØ	BITOBIT routine	Converts one byte of storage to printable EBCDIC values one bit at a time.
DSC (Model 6 and Model 10 Disk System only)		Phase ØC Parameter List	Contains additional COMMON entries used by Phase ØC.
DSD		Phase ØD Parameter List	Contains additional COMMON entries used by Phase ØD.
DSX		Execution Phases Parameter List	Contains additional COMMON entries used by Phases I, II, and III.

PHASE@ (in COMMON)

PHASE@ contains either the address of the current phase or the address of the next phase to be loaded. Immediately before control is to be passed to the next phase, PHASE@ is loaded with the address of where to load that phase. During execution of the current phase (before the point where control is to be passed to the next phase) PHASE@ contains the address of where the current executing phase was loaded.

To determine the next phase address, take the current phase address plus or minus the entry for that phase from the following list.

Current Phase	*Entry*
$DSORT	START@ + 256 + SYSINL *
	(+ Device Table length-Model 12 and Model 15)
$DSBA	(No change from previous phase Model 6 and Model 10) (Adjusted for shortened Device Table-Model 12 and Model 15)
$DSCZ	- SYSINL + TABLEL
$DSCB	+ $BUILDL_{temporary}$ + 256
$DSCC	- $BUILDL_{temporary}$ - 256 + $BUILDL_{final}$
$DSDA	(no change)
$DSEA	(no change)
$DSGA	(no change)
$DS1A,1B,1X	- BUILDL
	- Device Table length (Model 12 and Model 15)
$DS1Z	(no change)
$DS2A	(no change)
$DS3A	(no change)
$DS3S	(no change)

* For $DSORT, current PHASE@=FFFF

ERROR TABLE

The error table is built backwards by O.$DSCA; that is, the first entry is at a higher location in storage than the second entry. A 3-byte entry is placed in the table for each error found on the specification statements. The first two bytes contain the statement number where the error was found; the third byte contains the error number. BEGER1 in COMMON (COMMON+X'B4' through COMMON+X'B3') contains the address of the error table.

ALTERNATE COLLATING SEQUENCE TABLE

The alternate collating sequence table, if requested, is located in the last 256 bytes in main storage until Phase II and contains all the hexadecimal values. The table is modified by O.$DSBC according to the alternate collating sequence statements.

OLDS STRUCTURED ELEMENT ARRAY

The OLDS structured element array, built by $DS9G, contains information on a maximum of 18 old sort strings. OLDS@ in COMMON contains the address of this array. Each 16-byte entry contains:

Byte	*Contents*
0-2	Cylinder/sector/displacement of current block of string (CBLK)
3-4	Total number of records in block (REC#BT)
5-6	Number of records taken from block (REC#B)
7-8	Buffer address (BUF@)
9-10	Block address (BLK@)
11-12	Address of current record in block (REC@)
13-15	Cylinder/sector/displacement of next block of string(NBLK)

AVAIL TABLE

O.$DS9G builds the AVAIL table so that the first entry is at a higher location in storage than the second entry. The address of the table is found at @AVAIL in COMMON. Each 5-byte entry contains this information about the available blocks on disk:

Byte	*Contents*
0-2	Cylinder/sector/displacement of block
2-4	Total number of records in block (REC#BT)

DEVICE TABLE (MODEL 12 AND MODEL 15)

The Device Table is an 8-entry table built by $DSORT and contains information for each input file specified by a file statement. The first entry of the table represents INPUT or INPUT1 and the last entry represents INPUT8. During O.$DSORT and O.$DSBA each entry is 7 bytes long. O.$DSBA reformats the table stripping off DCODE and DRECL. This leaves each entry at a 4-byte length. The following is a description of a Device Table entry:

Information Byte

Byte	*Contents*	
0 (DINF0)	Bit 0	5444 file
	Bit 1	5445 file
	Bit 2	Tape file
	Bit 3	MFCU file*
	Bit 4	MFCM file*
	Bit 5	2501 file*
	Bit 6	1442 file*
	Bit 7	3741 file*

*Model 15 only

Device Attributes

Byte	*Contents-Tape File*	
1 (DATTR)	Bit 0	Fixed-record length
	Bit 1	Reserved
	Bit 2	Unblocked format
	Bit 3	Blocked format
	Bit 4	Reserved
	Bit 5	Reserved
	Bit 6	ASCII D.M. present
	Bit 7	Reel NL or NS
	Contents-Disk File	
	Bit 0	MVF offline
	Bit 1	MVF online
	Bit 2	File allocated
	Bits 3-7	Not used
	Contents-Card File (Model 15 only)	
	Bit 0	0 = Hopper 1 1 = Hopper 2
	Bits 1-7	Not used

Byte	*Contents-Tape File*
2-3 (DBLKL)	Block length from format 1 or record length for 3741 input (Model 15 only)
4 (DCODE)	Device code from format 1 (not present after O.$DSBA)
5-6 (DRECL)	Record length from format 1 (not present after O.$DSBA)

SUMMARY TABLE

This table, built by O.$DSCF, contains information about the type, length, and location of the summary fields within the work record. The table consists of 1 to 25 entries. If bit 3 of ATTRQ1 in COMMON is 1, then the first entry is that of the overflow indicator field. Otherwise, the first entry is the first summary data field. The entries contain:

Overflow Indicator

Byte	*Contents*
∅	XL1'FF' = Overflow indicator
1	Overflow indicator character (from Record character column)
2-3	Displacement of field from start of work record

First Summary Field

Byte	*Contents*
∅	XL1'ØØ' = Digit or unpacked decimal XL1'8∅' = Character (integer) XL1'4∅' = Packed decimal
1	Length minus 1 of field
2-3	Displacement of first field from start of work record (rightmost byte)

Following Summary Fields

Byte	*Contents*
∅	Type
1	Length minus 1 of field
2-3	Displacement of this field from previous field (rightmost byte)

TABLE OF WORK FILE EXTENTS

O.$DSBA builds this 24-byte table in COMMON. The high-order byte is MVFTBL. The first four bytes in the table contain four 1-byte device addresses (MVFQ1-MVFQ4), one address for each extent. The remainder of the table gives five bytes of information for each extent in this format:

Byte	*Contents*
0-1	5444 cylinder/sector of extent (MVFCS1-MVFCS4) or 5445 cylinder/head
2-4	Accumulated number of sectors with this extent

The last two bytes of each 5-byte entry are converted by O.$DSDA. Each entry then contains:

Byte	*Contents*
0-1	5444 cylinder/sector of extent (MVFCS1-MVFCS4) or 5445 cylinder/head
2-4	Accumulated number of work file blocks with this extent (MVF#1-MVF#4)

O.$DS9I BUFFER

This 271-byte buffer is required by the transient system SYSIN routine when the source statements are in the system reader. CARD@ in COMMON contains the address of the current statement located in this buffer.

O.$DS9W BUFFER

This 96-byte buffer stores the statement image from the JSWA Get routine when the source statements are on disk. CARD@ in COMMON contains the address of the current statement located in this buffer.

PHASES 1A, 1B, AND 1X BUFFERS

Work Buffer

The work buffer overlays the initialization portion of O.$DS1A, O.$DS1B, and O.$DS1X. The address of the work buffer is in WBUF@. The length of the work buffer is in WBUFL.

Input Buffer

The address of the input buffer is in IBUF@. The length of the buffer is in IBUFL.

Input Record Area

The input record area is used by O.$DS1A for the merge-selection internal sort. The address of this area is in IREC@. The length of the input record area is in IRECL or WRECL, whichever is greater.

PHASE II BUFFERS

Work Input Buffer

Phase II has several work input buffers, one for each order of merge. The first buffer overlays the initialization portion of Phase II. The address of this buffer is in IBUF@. The length of the buffer is in IBUFL.

Work Output Buffer

The work output buffer address is in WBUF@. The length of the buffer is in WBUFL. The records in the output buffer are placed on the work file.

PHASE III BUFFERS

Work Input Buffer

Phase III has three work input buffer areas. The first buffer overlays the initialization portion of Phase III. The address of the buffer is in IBUF@. The buffer length is in IBUFL.

Output File Buffer

The output file buffer contains the sorted records. These records are placed on the output file. The address of the output file buffer is in OBUF@. The length of the buffer is in OBUFL.

COPYRIGHT INFORMATION

This 33-byte area contains the program number for this program and copyright information as follows:

For the Model 6,
5703-SM1␣COPYRIGHT␣IBM␣CORP.␣1971

For the Model 10 Disk System,
5702-SM1␣COPYRIGHT␣IBM␣CORP.␣1970

For the Model 12,
5705-SM1␣COPYRIGHT␣IBM␣CORP.␣1976

For the Model 15,
5704-SM1␣COPYRIGHT␣IBM␣CORP.␣1974

The remainder of the area is filled with blanks. The copyright area follows the SYSIN routine for PHASE 0A.

PHASE IDENTIFICATION

This 5-byte area contains information for identifying the phase presently in storage. The contents of the area are:

Byte	*Contents*
Ø-3	Last four letters of the phase name
4	Program release number

The phase identification area is either in the first five bytes of the phase loaded into storage or eight bytes preceding the entry point to the phase.

COMMON -- Model 6 and Model 10 Disk System

COMMON is a 256-byte interphase table used by phases in both generation and execution portions of Disk Sort. COMMON is loaded into storage as the first 256 bytes of the Phase ØA load module. All length fields in COMMON are initialized as hexadecimal zeros and all address fields are initialized as hexadecimal F's when Phase ØA is assembled.

Figure 5-1 is a storage map showing the general data areas in COMMON. Figure 5-2 describes each field in COMMON for the Model 6 and Model 10 Disk System. A brief description of each of the general data areas shown in Figure 5-1 follows.

$DSC (X'ØØ'-X'Ø3'): The first four bytes of COMMON contain C'$DSC'. This field serves as an identification field for Disk Sort COMMON.

Counter Information (X'Ø4'-X'Ø9'): This area counts the blocks written on the work file.

Attribute Information Area (X'0A'-'0F'): This area contains attribute information about the 5445 Disk Sort Feature, 7-track tape,and Summary Sort.

Error Information (X'1Ø'-X'11'): This is the logging area for Disk Sort errors.

Job Environment Data (X'12'-X'4Ø'): This area contains data about the system environment within which the program will operate.

Job Attributes Data (X'41'-X'68'): This area contains information describing the specific job to be performed by the program.

General Status Information (X'69'-X'71'): This area tells which phase of Disk Sort is being executed and what type of errors have been detected.

Generation Phases Work Area (X'72'-X'82'): This area contains information used only by the generation phases of Disk Sort. It is overlaid by the disk I-O status work areas during the execution portion of the program.

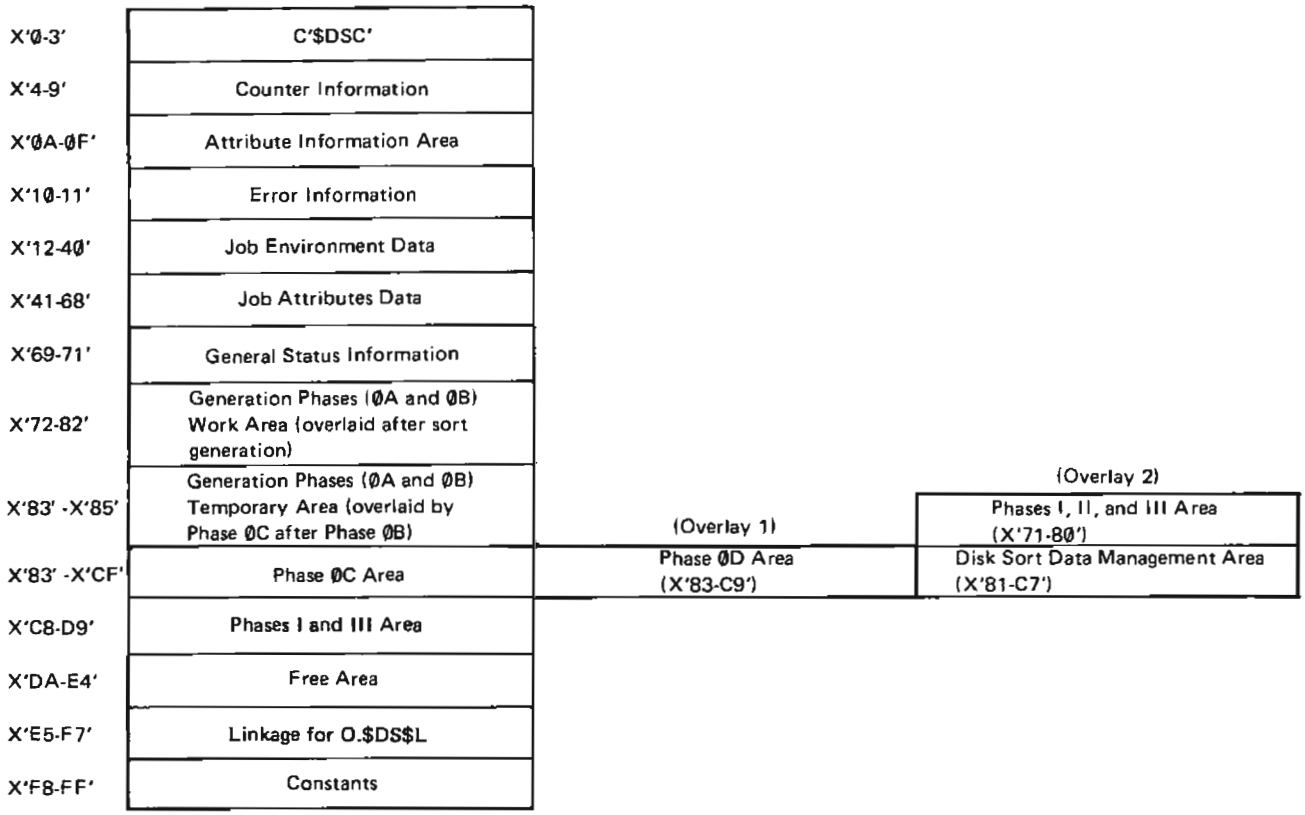


Figure 5-1. Storage Map of COMMON

Generation Phase Temporary Work Area (X'83'-X'85'): This area contains information concerning the output file size. It is overlaid after Phase ØB.

Phase ØC Area (X'83'-X'CF'): This area contains fields used only by Phase ØC.

Phase ØD Area (X'83'-X'C9'): This area contains fields used only by Phase ØD.

Phases I-III Area (X'71'-X'8Ø'): This area contains fields used by Phases I-III.

Disk Sort Data Management Area (X'81'-X'C7'): This area contains fields used by Disk Sort data management.

Phases 1A, 1B, 1X, and III Area (X'C8'-X'D9'): This area contains fields used only by Phase 1A, 1B, 1X, or III.

Free Area (X'DA'-X'E4'): This is a free area in COMMON.

O.DSL Linkage (X'E5'-X'F7'): This area contains instructions to load and give control to O.DSL when a terminal dump of COMMON is requested.

Constants (X'F8'-X'FF'): This area contains constants that are used by both the generation and execution portions of Disk Sort.

Name	Hexadecimal Displacement	Bytes	Description
$DSC	Ø3	Ø-3	Character '$DSC', the ID for COMMON
Counter Information			
WBLK#	Ø6	4-6	Work block counter in current phase
WBLK#T	Ø9	7-9	Original (Phase I) counter
Attribute Information Area			
DATA1	ØA	1Ø	Bit Ø 7-track tape input with converter Bit 1 7-track tape input with translator Bit 2 7-track tape output with converter Bit 3 7-track tape output with translator Bit 4 5445 feature not available Bits 5-7 Not used
DATA2	ØB	11	Bit 0 Use new cumulative length for SORTRS Bits 1-4 Not used Bit 5 Output tape is labeled Bit 6 Not used Bit 7 Output file size check valid in O.$DS1Z; if bit is off, check is valid
OREC	ØD	12-13	Tape output record length in Format-1
TPEIN	ØE	14	Attributes for tape input file
TPEOUT	ØF	15	Attributes for tape output file
Error Information			
ERRQ	1Ø	16	Error byte used by O.$DS1Z and O.$DS4A
SPVECT	11	16-17	Vector code for sort routines
ERR2HQ	11	16-17	Error byte used for internal sort errors '2H' halt error indicator (Model 10 Disk System) 'CD45' halt error indicator (Model 6)

Figure 5-2 (Part 1 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement		Bytes	Description
Job Environment Data				
ENVQ1	12	18	Bit Ø	Force O.$DS1A sort.algorithm
			Bit 1	Force O.$DS1B sort algorithm
			Bit 2	Force O.$DS1X sort algorithm
			Bit 3	First 5445 input is on D1; if off, first 5445 input is on D2
			Bit 4	First 5445 output is on D1
			Bit 5	First 5445 output is on D2
			Bit 6	D1 is available for auto-allocate
			Bit 7	D2 is available for auto-allocate
ENVQ2	13	19	Bit Ø	Source statements
			Bit 1	Source statements from SYSIN reader
			Bit 2	Source statements from scheduler work area
			Bit 3	Source statements from source library
			Bit 4	DEBUG specified on header statement (dump requested)
			Bit 5	Not used
			Bit 6	Work file statement present
			Bit 7	In DPF level2
ENVQ3	14	2Ø	Bit Ø	Multivolume input file
			Bit 1	Multivolume work file (standard setting of 1)
			Bit 2	Multivolume output file
			Bit 3	Not used
			Bit 4	Output on upper drive if Ø
			Bit 5	R1 available for work space if Ø
			Bit 6	R2 available for work space if Ø
			Bit 7	F2 available for work space if Ø
FILE1	15	21	Bit Ø	5444 input
			Bit 1	5445 input
			Bit 2	Tape input
			Bit 3	5444 work
			Bit 4	5445 work
			Bit 5	5444 output
			Bit 6	5445 output
			Bit 7	Tape output
FILE2	16	22	Bit Ø	ASCII input
			Bit 1	ASCII output
			Bit 2	Write to work file not verified (non-verify option)
			Bit 3	Variable record format—O.$DSBA issues error message.
			Bit 4	Error in tape OCL for Disk Sort
			Bit 5	Input is on 7-track tape; if off, input is on 9-track tape
			Bit 6	Output is on 7-track tape; if off, output is on 9-track tape
			Bit 7	Not used

Figure 5-2 (Part 2 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
PHASE@	18	23-24	Next phase load address (see *Phase Address Computation Table*)
TABLEL	1A	25-26	Length of summary table
START@	1C	27-28	Address of the first byte of program level
COREL	1E	29-30	Length of the program level in bytes
ALTSQ@	20	31-32	Address of the Alternate Collating Sequence routine
ALTSQL	22	33-34	Length of the Alternate Collating Sequence routine in bytes (defaults to X'0000')
BUILD@	24	35-36	Address of the Select/Build routine
BUILDL	26	37-38	Length of the Select/Build routine in bytes (defaults to X'0000')
IREC@	28	39-40	Address of the current input record for the Select/Build routine
MVFTBL	29	41	Table of work file extents
MVFQ1	29	41	Q-byte for extent number 1
MVFQ2	2A	42	Q-byte for extent number 2
MVFQ3	2B	43	Q-byte for extent number 3
MVFQ4	2C	44	Q-byte for extent number 4
MVFCS1	2E	45-46	Cylinder/sector of first extent
MVF#1	31	47-49	Number of blocks in first extent
MVFCS2	33	50-51	Cylinder/sector of second extent
MVF#2	36	52-54	Accumulated number of blocks with second extent
MVFCS3	38	55-56	Cylinder/sector of third extent
MVF#3	3B	57-59	Accumulated number of blocks with third extent
MVFCS4	3D	60-61	Cylinder/sector of fourth extent
MVF#4	40	62-64	Accumulated number of blocks with fourth extent

Figure 5-2 (Part 3 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
Job Attributes Data			
ATTRQ1	41	65	Bit Ø SORTA Bit 1 SORTR or SORTRS Bit 2 SORTRS Bit 3 Overflow indicator Bit 4 O.$DSCB has messages to print Bit 5 Not used Bit 6 Output is a deferred mount Bit 7 Last pass criteria Ø = STR# ≤ OM 1 = STR# < OM
ATTRQ2	42	66	Bit Ø Ø = Ascending sequence 1 = Descending sequence Bit 1 At lease one valid alternate collating sequence found Bit 2 Alternate collating sequence: Ø = EBCDIC sequence 1 = Special sequence Bit 3 Drop key Bit 4 Include all Bit 5 Continuation of next statement assumed Bit 6 Field statement for current set Bit 7 Ø = Record type specified 1 = Record type assumed
ATTRQ3	43	67	Bit Ø Summary table built (SORTRS) Bit 1 Pack-Include/Omit routine needed Bit 2 Pack-Field routine needed Bit 3 First specification in set Bit 4 Last statement type not saved Bit 5 Page sequence not checked Bit 6 Data statement in current set Bit 7 Generating of Select/Build routine stopped
PRTOPN	44	68	Print option (defaults to C'Ø')
IRECL	46	69-7Ø	Input record length
KEYL	47	71	Work record key length-1
WRECL	49	72-73	Work record length
ORECLH	4B	74-75	Output record length specified on header statement
KEY2L	4D	76-77	Control field length-1
ORECLP	4F	78-79	Output record length for Phase III
FLDCNT	5Ø	8Ø	Number of entries in summary table

Figure 5-2 (Part 4 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
IBUFL	52	81-82	Total length of input buffer
IBUF#	53	83	Number of input buffers
IBLKL	55	84-85	Length of one input block
ISTYPE	56	86	Internal sort type: E = Exchange sort used by Phase 1A S = Selection sort used by Phase 1B with module $DS1S X = Tournament sort used by Phase 1X
ISL	58	87-88	Length of internal sort area
ISG	5A	89-90	Number of records in internal sort area
WBF	5C	91-92	Normal work blocking factor
WBLKL	5E	93-94	Length of normal work block
WBUFL	60	95-96	Total length of normal work buffer
OM	61	97	Order of merge (Phase II)
OM3	62	98	Order of merge (Phase III)
OBUFL	63-64	99-100	Total length of output buffer
OBUF#	65	101	Number of output buffers
OBLKL	66-67	102-103	Length of one output block
SPLITL	68	104	Number of tracks in a split cylinder
General Status Information			
$PHASE	6A	105-106	Current phase in EBCDIC
IOERRQ	6B	107	Bit 0 SYSIN error Bit 1 Scheduler work area read error Bit 2 Source member read error Bit 3 SYSLOG error Bit 4 Disk error (input file) Bit 5 Disk error (work file read) Bit 6 Disk error (work file write) Bit 7 Disk error (output file)
WRNERQ	6C	108	Bit 0 Warning error in Phase 0A Bit 1 Warning error in Phase 0B Bit 2 Warning error in Phase 0C Bit 3 Warning error in Phase 0D Bits 4-6 Not used Bit 7 Statement sequence errors

Figure 5-2 (Part 5 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
SEVERQ	6D	109	Bit Ø Severe error in Phase ØA Bit 1 Severe error in Phase ØB Bit 2 Severe error in Phase ØC Bit 3 Severe error in Phase ØD Bit 4 Severe error in Phase ØF Bit 5 Severe error in Phase I Bit 6 Severe error in Phase II Bit 7 Severe error in Phase III
TMLERQ	6E	110	Bit Ø Terminal error in Phase ØA Bit 1 Terminal error in Phase ØB Bit 2 Terminal error in Phase ØC Bit 3 Terminal error in Phase ØD Bit 4 Terminal error in Phase ØF Bit 5 Terminal error in Phase I Bit 6 Terminal error in Phase II Bit 7 Terminal error in Phase III
WRNER#	6F	111	Warning error counter
SEVER#	70	112	Severe error counter
TMLER#	71	113	Terminal error counter
Generation Phases Work Areas (overlaid after sort generation)			
SETBL@	73	114-115	Address of the error table
NETBL@	75	116-117	Address of the next error table entry
CARD@	77	118-119	Address of the statement image in SYSIN routine (high-order byte)
LINE#	78	120	Line number on a page for generation phase
STMT#	7C	122-124	Current statement number in decimal
SYSIN@	7E	125-126	Address of SYSIN routine
SYSINL	80	127-128	Length of the SYSIN routine in bytes
TUILD@	82	129-130	Temporary address of the Select/Build routine
PHO$$$	82	130	Last byte of constant area used by the generation phases
OUTF1#	85	131-133	Number of output records for the disk file
Phase ØC Area (overlays Phases ØA and ØB Area)			
CARD	83	131	Columns 1-39 of current statement
PGLN1	87	132-135	Page line
TYPE1	88	136	Card type

Figure 5-2 (Part 6 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
TYPEF	89	137	Type of control field
CONT1	89	137	Continuation record
CZDP1	8A	138	C/Z/D/P/U
FC1FB1	8B	139	Factor 1 begin column (from)
FC1FE1	8E	140-142	Factor 1 end column (from)
FC1TB1	8F	143	Factor 1 begin column (to)
FC1TE1	92	144-146	Factor 1 end column (to)
RECHA1	93	147	Record character
SUBCH1	94	148	Substitute character
REL1	94	148	Relationship
CONTF	95	149	Forced continuation
FIOCT1	95	149	Field or constant
CONBG1	96	150	Beginning of constant area
FC2FB1	96	150	Factor 2 from begin column
NEWSUM	98	152	New resultant length of summary field
FC2FE1	99	151-153	Factor 2 from end column
FC2TB1	9A	154	Factor 2 to begin column
FC2TE1	9D	155-157	Factor 2 to end column
TLGTH1	9E	158	Temporary length
CONEN1	A9	159-169	End of constant area
DISP21	AB	170-171	Displacement of operand 2
LNGTH1	AD	172-173	Length for move
CWADP1	AE	174	Displacement in Build control word
DISP11	B0	175-176	Displacement of operand 1
HEXNO1	B2	177-178	Hex work field
BEGER1	B4	179-180	Beginning error table
SAVER1	B6	181-182	End of error table

Figure 5-2 (Part 7 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
LNGTH2	B8	183-184	Save area for length
IWADP1	BA	185-186	Permanent displacement in control word
KEYTL	BC	187-188	Test for control field length
DSCA5@	BE	189-190	Address of #DSCA5 in O.$DSCA
SUMTB@	BE	189-190	Address of summary table
DSCA6@	C0	191-192	Address of #DSCA6 in O.$DSCA
DSCA8@	C2	193-194	Address of #DSCA8 in O.$DSCA
CONST@	C4	195-196	Address constants in O.$DSCA
CASW1	C5	197	Bit 0 Set XR1 = XR2 Bit 1 Force sequence Bit 2 Summary V found Bit 3 Displacement is -256 Bit 4 Drop key: no data or summary specifications Bit 5 SORTRS: resultant length feature Bit 6 Not SORTRS, but S card Bit 7 Too many errors
CASW2	C6	198	Not used
CCDLN1	C8	199-200	Work field to check work record length
WRECD	CA	201-202	Current displacement in work record
OFLOW@	CC	203-204	Overflow table entry
NSUM@	CE	205-206	Next summary table entry
SUM#	CF	207	Summary table element counter
SUMI	83	128-134	Summary table element
Phase 0D Area (overlays Phase 0C equates)			
ERRORQ	83	131 00	No errors
		01	Phase I main storage assigned is too small
		02	Phase I main storage too small for minimum WBUFL
		03	Phase II main storage too small for minimum merge
		04	Phase III main storage too small for minimum WBUFL, OM=2
		06	For OM=2, WRECL > WBUFL
		07	For OM=4, WRECL > WBUFL
		08	Phase I main storage too small for WBUFL, OM=4
		09	Phase III main storage too small for minimum buffer
		0A	Attempted to force O.$DS1B, but not enough room
		10	5445 used, but feature not available
		20	Tape used, but support not available

Figure 5-2 (Part 8 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
PGML1A	85	132-133	O.$DS1A active length
PGML1B	87	134-135	O.$DS1B active length
PGML1S	89	136-137	O.$DS1S active length
PGML1X	8B	138-139	O.$DS1X active length
PGML2A	8D	140-141	O.$DS2A active length
PGML3A	8F	142-143	O.$DS3A active length
FIXL1	91	144-145	Fixed Phase I length
RSAL1A	93	146-147	Remaining record storage area for Phase 1A
RSAL1S	95	148-149	Remaining record storage area for Phase 1B
RSAL1X	97	150-151	Remaining record storage area for Phase 1X
RSAL2A	99	152-153	Remaining record storage area for Phase II
MTOD#1	9B	154-155	Remaining record storage area for Phase III
RSAS2A	9C	156	Record storage area in sectors for Phase II
MINIL	9E	157-158	Minimum input buffer length
MINOL	A0	159-160	Minimum output buffer length
MINISL	A2	161-162	Minimum sort work area
MINISG	A4	163-164	Minimum number of records in sort area
MINISB	A6	165-166	Additional minimum sort work area
MINWL	A8	167-168	Minimum work buffer length
MIN2L	AA	169-170	Minimum Phase II (III) record storage area length
IRECLP	AC	171-172	Input record area length
WBUFLP	AE	173-174	Effective work buffer length
WBUFS	AF	175	Work buffer length in sectors
WBLKLP	B1	176-177	Effective work block length
WBLKS	B2	178	Work block length in sectors
SORTRL	B4	179-180	Internal sort record length

Figure 5-2 (Part 9 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
$OM	B5	181	Order of merge + 1
OMA	B5	181	Actual usable order of merge
OMQ	B6	182	Order of merge search switch
TOP	B9	183-185	Scratch area
$$$$	B9	185	Working storage
AVAIL	BB	186-187	Current available storage in phase
MTL1	BD	188-189	Unused storage Phase I
MTL2	BF	190-191	Unused storage Phase II
MTL3	C1	192-193	Unused storage Phase III
ADDG	C3	194-195	Additional internal sort records
ADDL	C5	196-197	Space used by ADDG
ADDOL	C7	198-199	Additional output buffer space
IBF	C9	200-201	Input blocking factor
Phases I, II, and III Area			
NXTD8@	73	114-115	Address Update routine
@POP	75	116-117	Physical Read/Write routine
@$DS9P	77	118-119	Logical Write routine
@$DS9G	79	120-121	Logical Read routine
@DTAMT	7B	122-123	Data input/output
MOVER	7C	124	First byte of the fixed branch to O.$DSZA
MOVERR	7E	125-126	Last byte of the fixed branch to O.$DSZA
MOVER@	80	127-128	Move routine address
Disk Sort Data Management Areas			
@WKDTF	82	129-130	Work file DTF
WETBL@	84	131-132	Current work file extent table entry
OLDS@	86	133-134	Address of the old string table

Figure 5-2 (Part 10 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
I	88	135-136	String number index times 16
INQ	89	137	WGETL master switch: F = First call of pass I = Intermediate call N = Start of new string merge
OMP	8A	138	Order of merge for pass
OMS	8B	139	Order of merge for current new string
NSTR	8E	140-142	Cylinder/sector/displacement of the first block in the next old string
READ@	90	143-144	Storage address of current read
READ#	91	145	Number of sectors for current read
@AVAIL	93	146-147	Address of the high-order byte of the available list
AVAIL@	95	148-149	Address of the high-order byte of the current available element
IREC#T	98	150-152	Total number of input records
PASS	99	153	Current pass number
PASS#T	9A	154	Total number of expected passes
STR#	9C	155-156	Current string counter
STR#1	9F	157-159	Cylinder/sector/displacement of the first block of the first string
STR#T	A1	160-161	Number of strings in last pass
WBLK@	A3	162-163	Address of the current work block
WBUF@	A5	164-165	Address of the work buffer area
OBUF@	A5	164-165	Address of output buffer
WORK1	A7	166-167	2-byte work area
WORK2	A9	168-169	2-byte work area
WBLKQ	AA	170	WPUTL block type: N = Start new string O = Continue old string
WCBLKC	AD	171-173	Cylinder/sector/displacement of current work block
WCSTRC	B0	174-176	Cylinder/sector/displacement of current work string

Figure 5-2 (Part 11 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
WLSTRC	B3	177-179	Cylinder/sector/displacement of last work string
WNBLKC	B6	180-182	Cylinder/sector/displacement of next work block
WORKQ	B7	183	WPUTL request switch: F = First request I = Intermediate request L = Last request
WREC@	B9	184-185	Address of the current located work record for output
WREC#	BC	186-188	Work record counter
WREC#T	BF	189-191	Total number of work records created
WRECQ	C0	192	WPUTL record position: F = Next record is first in block I = Last record is not last in block L = Last record is last in block
WRITE@	C2	193-194	Storage address of next write
WRITE#	C3	195	Number of sectors for next write
WRPBC	C5	196-197	WPUTL record and block counter
WSTRQ	C6	198	WPUTL string continuation switch: C = Continue current string E = End of current string
BUILDQ	C7	199	Work record built switch: N = No Y = Yes
Phase I Areas			
LIREC@	C9	200-201	Original address of the located input record
BREC@	CB	202-203	Address of the high-order byte of the work record build location
SREC@	CD	204-205	Address of the winning work record
SORTQ	CE	206	Master internal switch: F = First call I = Intermediate call L = Last call
SRECQ	CF	207	Winning record type: D = Dummy sort E = End sort I = Intermediate

Figure 5-2 (Part 12 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

Name	Hexadecimal Displacement	Bytes	Description
SSTRQ	DØ	2Ø8	Sort string switch: C = Continue old string N = New string
CG#	D2	2Ø9-21Ø	Number of records in internal sort area's current string
ISORT@	D4	211-212	Address of the high-order byte of the internal sort area for Phase 1B
MVTBL	D9	213-217	Instruction to move summary table (except op code)
Free Area	E4	218-228	Not used
Phase III Areas			
WORK	C9	200-201	Work Area
AVAIL	CB	202-203	Current available storage in phase
MINOL	CD	204-205	Minimum output buffer length
ERRORQ	CE	206	Error switch
ADDOL	D0	207-208	Additional output buffer space
MTL3	D2	209-210	Unused storage—Phase III
OREC#	D5	211-213	Output record counter
Free Area	E4	214-228	Not used
O.DSL Linkage			
XCTL$L	F7	229-247	Instructions to load and give control to O.DSL when a terminal dump of COMMON is requested
Constants			
I0000	F9	248-249	XL2'0000'
I0001	FB	250-251	XL2'0001'
I0002	FD	252-253	XL2'0002'
IFFFF	FF	254-255	XL2'FFFF'

Figure 5-2 (Part 13 of 13). Field Description of COMMON (Model 6 and Model 10 Disk System)

COMMON — Model 12 and Model 15

COMMON is a 256-byte interphase table used by phases in both generation and execution portions of Disk Sort. COMMON is loaded into storage as the first 256 bytes of the Phase 0A load module. All length fields in COMMON are initialized as hexadecimal zeros and all address fields are initialized as hexadecimal F's when Phase 0A is assembled.

Figure 5-3 is a storage map showing the general data areas in COMMON. Figure 5-4 describes each field in COMMON for the Model 12 and Model 15. A brief description of each of the general data areas shown in Figure 5-3 follows.

$DSC (X'00'-X'03'): The first four bytes of COMMON contain C'$DSC'. This field serves as an identification field for Disk Sort COMMON.

Counter Information (X'04'-X'09'): This area counts the blocks written on the work file.

Attribute Information Area (X'0A'-'0F'): This area contains attribute information about the 5445 Disk Sort Feature, 7-track tape, and Summary Sort.

Error Information (X'10'-X'11'): This is the logging area for Disk Sort errors.

Job Environment Data (X'12'-X'53'): This area contains data about the system environment within which the program will operate.

Job Attributes Data–Part 1 (X'54'-X'68'): This area contains information describing the specific job to be performed by the program.

General Status Information (X'69'-X'71'): This area tells which phase of Disk Sort is being executed and what type of errors have been detected.

Generation Phases Work Area (X'72'-X'82'): This area contains information used only by the generation phases of Disk Sort. It is overlaid by the disk I-O status work areas during the execution portion of the program.

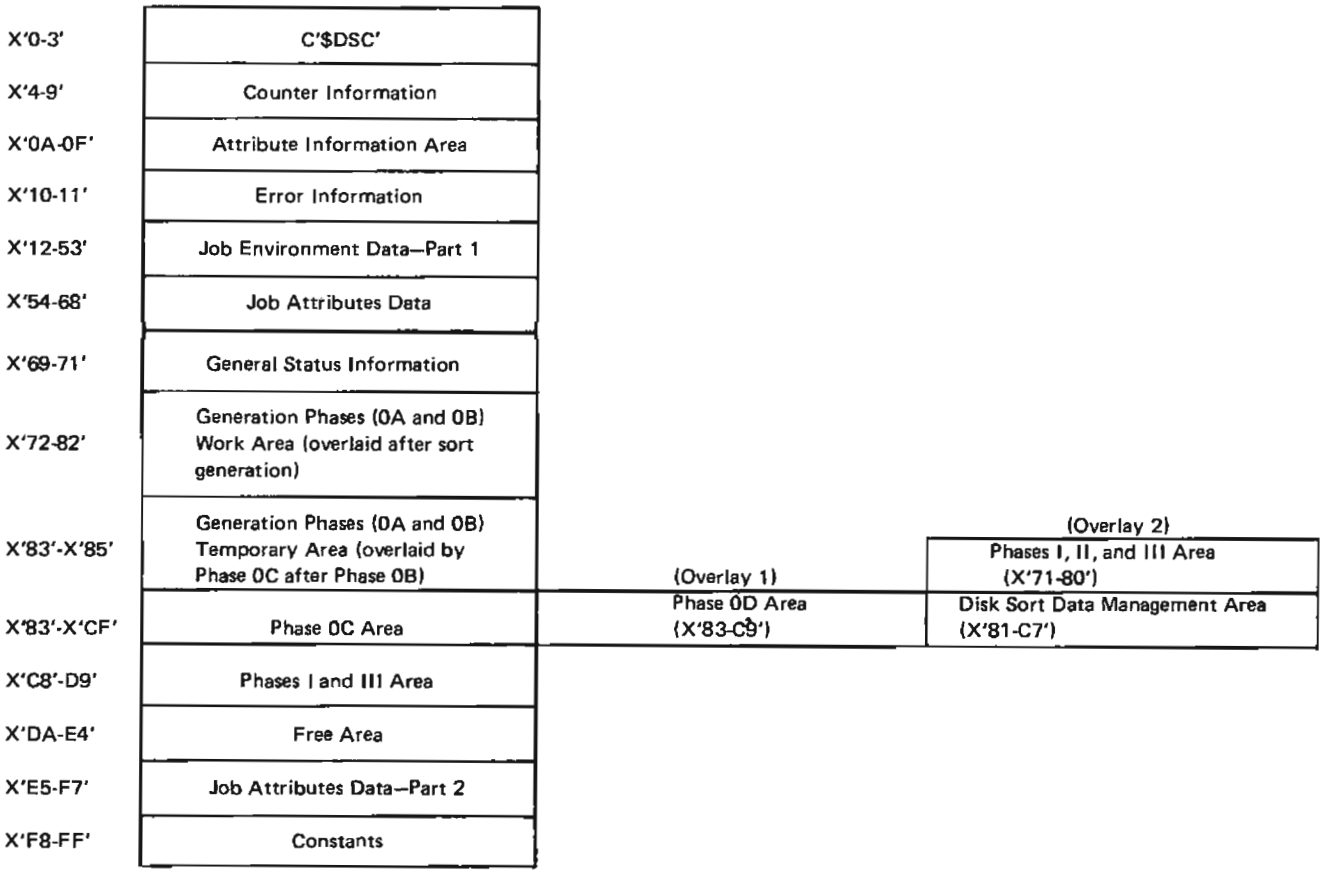


Figure 5-3. Storage Map of COMMON

Generation Phase Temporary Work Area (X'83'-X'85'): This area contains information concerning the output file size. It is overlaid after Phase 0B.

Phase 0C Area (X'83-X'CF'): This area contains fields used only by Phase 0C.

Phase 0D Area (X'83'-X'C9'): This area contains fields used only by Phase 0D.

Phases I-III Area (X'71'-X'80'): This area contains fields used by Phases I-III.

Disk Sort Data Management Area (X'81'-X'C7'): This area contains fields used by Disk Sort data management.

Phases 1A, 1B, 1X, and III Area (X'C8'-X'D9'): This area contains fields used only by Phase 1A, 1B, 1X, or III.

Free Area (X'DA'-X'E4'): This is a free area in COMMON.

Job Attributes Data–Part 2 (X'E5'-X'F7'): This area contains information describing the job to be performed by the program.

Constants (X'F8'-X'FF'): This area contains constants that are used by both the generation and execution portions of Disk Sort.

Name	Hexadecimal Displacement	Bytes	Description
$DSC	03	0-3	Character '$DSC', the ID for common
Counter Information			
WBLK#	06	4-6	Work block counter in current phase
WBLK#T	09	7-9	Original (Phase I) counter
Attribute Information Area			
DATA1	0A	10	Bit 0 7-track tape input with converter Bit 1 7-track tape input with translator Bit 2 7-track tape output with converter Bit 3 7-track tape output with translator Bit 4 Output file on D1 Bit 5 Output file on D2 Bit 6 Output file on D3 (Model 15 only) Bit 7 Output file on D4 (Model 15 only)
DATA2	0B	11	Bit 0 96-column card input (Model 15 only) Bit 1 80-column card input (Model 15 only) Bit 2 Multiple tape files Bit 3 Multiple 5444 input files Bit 4 Multiple 5445 input files Bit 5 Output tape is labeled Bit 6 Work file is split/cylinder (Model 15 only) Bit 7 Output file size check valid in O.$DS1Z; if bit is off check is valid
OREC	0D	12-13	Tape output record length in Format-1
OUTTK#	0D	13	Number of tracks given for disk output
TKSIZE	0E	14	Sectors/track for the 5445 files
TPEOUT	0F	15	Attributes for tape output file
Error Information			
ERRQ	10	16	Error byte used by O.$DS1Z and O.$DS4A
SPVECT	11	16-17	Vector code for sort routines
ERR2HQ	11	16-17	Error byte used for internal sort errors '2H' halt error indicator

Figure 5-4 (Part 1 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement		Bytes	Description
Job Environment Data				
ENVQ1	12	18	Bit 0	D1 Available for work space
			Bit 1	D2 Available for work space
			Bit 2	D3 Available for work space (Model 15 only)
			Bit 3	D4 Available for work space (Model 15 only)
			Bits 4-5	Not used
			Bits 6-7	01 = Force O.$DS1A algorithm
				10 = Force O.$DS1B algorithm
				11 = Force O.$DS1X algorithm
ENVQ2	13	19	Bit 0	Source statements
			Bit 1	Source statements from SYSIN reader
			Bit 2	Source statements from scheduler work area
			Bit 3	Source statements from source library
			Bit 4	DEBUG specified on header statement (dump requested)
			Bit 5	No used
			Bit 6	Work file statement present
			Bit 7	In DPF partition 2
ENVQ3	14	20	Bit 0	Multivolume input file
			Bit 1	Multivolume work file (standard setting of 1)
			Bit 2	Multivolume output file
			Bit 3	Not used
			Bit 4	Output on upper drive if 0
			Bit 5	R1 available for work space if 0
			Bit 6	R2 available for work space if 0
			Bit 7	F2 available for work space if 0
FILE1	15	21	Bit 0	5444 input
			Bit 1	5445 input
			Bit 2	Tape input
			Bit 3	5444 work
			Bit 4	5445 work
			Bit 5	5444 output
			Bit 6	5445 output
			Bit 7	Tape output
FILE2	16	22	Bit 0	ASCII input
			Bit 1	ASCII output
			Bit 2	Write to work file not verified (non-verify option)
			Bit 3	Variable record format—O.$DSBA issues error message.
			Bit 4	Error in tape OCL for Disk Sort
			Bit 5	Input is on 7-track tape; if off, input is on 9-track tape
			Bit 6	Output is on 7-track tape; if off, output is on 9-track tape
			Bit 7	Not used

Figure 5-4 (Part 2 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
PHASE@	18	23-24	Next phase load address (see *Phase Address Computation Table)*
TABLEL	1A	25-26	Length of summary table
START@	1C	27-28	Address of the first byte of partition
COREL	1E	29-30	Length of the partition in bytes
ALTSQ@	20	31-32	Address of the Alternate Collating Sequence routine
ALTSQL	22	33-34	Length of the Alternate Collating Sequence routine in bytes (defaults to X'0000')
BUILD@	24	35-36	Address of the Select/Build routine
BUILDL	26	37-38	Length of the Select/Build routine in bytes (defaults to X'0000')
IREC@	28	39-40	Address of the current input record for the Select/Build routine
MVFTBL	29	41	Table or work file extents
MVFQ1	29	41	Q-byte for extent number 1
MVFQ2	2A	42	Q-byte for extent number 2
MVFQ3	2B	43	Q-byte for extent number 3
MVFQ4	2C	44	Q-byte for extent number 4
MVFCS1	2E	45-46	Cylinder/sector of first extent
MVF#1	31	47-49	Number of blocks in first extent
MVFCS2	33	50-51	Cylinder/sector of second extent
MVF#2	36	52-54	Accumulated number of blocks with second extent
MVFCS3	38	55-56	Cylinder/sector of third extent
MVF#3	3B	57-59	Accumulated number of blocks with third extent
MVFCS4	3D	60-61	Cylinder/sector of fourth extent
MVF#4	40	62-64	Accumulated number of blocks with fourth extent
DTAB@	42	65-66	Address of Device Table
INPUT#	43	67	1 byte input file counter: x '40'— 1st file = INPUT x 'F1'— 1st file = INPUT1 x 'FF'— 1st file not present

Figure 5-4 (Part 3 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
INDTF@	45	68-69	Address of DTF area
DMLD@	47	70-71	Address of Data Management load point
DMLNG	48	72	Length of Data Management in sectors
IBUF@	4A	73-74	Address of input buffer-Phase I
SW1M1	4B	75	** SWITCH ** Bit 0 1st call of O.$DS1M Bit 1 1st file name = INPUT Bit 2 Card device and IRECL > 80 (Model 15 only) Bit 3 Halt '26' same device Bit 4 Skip file-1 option-'26' Bits 5-7 Not used
CRECL	4D	77	Record length of file currently being processed
XTRA	51	78-81	Not used
LINE#	52	82	Current line number
#MAXL	53	83	System page size
Job Attributes Data – Part 1			
ATTRQ1	54	84	Bit 0 SORTA Bit 1 SORTR or SORTRS Bit 2 SORTRS Bit 3 Overflow indicator Bit 4 O.$DSCB has messages to print Bit 5 Not used Bit 6 Output is a deferred mount Bit 7 Last pass criteria 0 = STR# ≤ OM 1 = STR# ≤ OM
ATTRQ2	55	85	Bit 0 0 = Ascending sequence 1 = Descending sequence Bit 1 At lease one valid alternate collating sequence found Bit 2 Alternate collating sequence: 0 = EBCDIC sequence 1 = Special sequence Bit 3 Drop key Bit 4 Include all Bit 5 Continuation of next statement assumed Bit 6 Field statement for current set Bit 7 0 = Record type specified 1 = Record type assumed

Figure 5-4 (Part 4 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
ATTRQ3	56	86	Bit 0 Summary table built (SORTRS) Bit 1 Pack-Include/Omit routine needed Bit 2 Pack-Field routine needed Bit 3 First specification in set Bit 4 Last statement type not saved Bit 5 Page sequence not checked Bit 6 Data statement in current set Bit 7 Generating of Select/Build routine stopped
PRTOPN	57	87	Print option (defaults to C'0')
IRECL	59	88-89	Input record length
KEYL	5A	90	Work record key length-1
WRECL	5C	91-92	Work record length
ORECLH	5E	93-94	Output record length specified on header statement
KEY2L	60	95-96	Control field length-1
RSAL3A	62	97-98	Remaining record storage area for Phase III
FLDCNT	63	99	Number of entries in summary table
IBUFL	65	100-101	Total length of input buffer
IBUF#	66	102	Number of input buffers
IBLKL	68	103-104	Length of one input block
General Status Information			
$PHASE	6A	105-106	Current phase in EBCDIC
IOERRQ	6B	107	Bit 0 SYSIN error Bit 1 Scheduler work area read error Bit 2 Source member read error Bit 3 SYSLOG error Bit 4 Disk error (input file) Bit 5 Disk error (work file read) Bit 6 Disk error (work file write) Bit 7 Disk error (output file)
WRNERQ	6C	108	Bit 0 Warning error in Phase 0A Bit 1 Warning error in Phase 0B Bit 2 Warning error in Phase 0C Bit 3 Warning error in Phase 0D Bits 4-6 Not used Bit 7 Statement sequence errors

Figure 5-4 (Part 5 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
SEVERQ	6D	109	Bit 0 Severe error in Phase 0A Bit 1 Severe error in Phase 0B Bit 2 Severe error in Phase 0C Bit 3 Severe error in Phase 0D Bit 4 Severe error in Phase 0F Bit 5 Severe error in Phase I Bit 6 Severe error in Phase II Bit 7 Severe error in Phase III
TMLERQ	6E	110	Bit 0 Terminal error in Phase 0A Bit 1 Terminal error in Phase 0B Bit 2 Terminal error in Phase 0C Bit 3 Terminal error in Phase 0D Bit 4 Terminal error in Phase 0F Bit 5 Terminal error in Phase I Bit 6 Terminal error in Phase II Bit 7 Terminal error in Phase III
WRNER#	6F	111	Warning error counter
SEVER#	70	112	Severe error counter
TMLER#	71	113	Terminal error counter
Generation Phases Work Areas (overlaid after sort generation)			
SETBL@	73	114-115	Address of the error table
NETBL@	75	116-117	Address of the next error table entry
CARD@	77	118-119	Address of the statement image in SYSIN routine (high-order byte)
INFO	78	120	Bit 0 MFCU input Bit 1 MFCM input Bit 2 2501 input Bit 3 1442 input Bit 4 3741 input Bits 5-7 Not used (Model 15 only)
STMT#	7C	121-124	Current statement number in decimal
SYSIN@	7E	125-126	Address of SYSIN routine
SYSINL	80	127-128	Length of the SYSIN routine in bytes
TUILD@	82	129-130	Temporary address of the Select/Build routine
PHO$$$	82	130	Last byte of constant area used by the generation phases
OUTF1#	86	134	Temporary area for output file size-in records or tracks (Phases 0A and 0B)

Figure 5-4 (Part 6 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
Phase 0C Area (overlays Phases 0A and 0B Area)			
CARD	83	131	Columns 1-39 of current statement
PGLN1	87	132-135	Page line
TYPE1	88	136	Card type
TYPEF	89	137	Type of control field
CONT1	89	137	Continuation record
CZDP1	8A	138	C/Z/D/P/U
FC1FB1	8B	139	Factor 1 begin column (from)
FC1FE1	8E	140-142	Factor 1 end column (from)
FC1TB1	8F	143	Factor 1 begin column (to)
FC1TE1	92	144-146	Factor 1 end column (to)
RECHA1	93	147	Record character
SUBCH1	94	148	Substitute character
REL1	94	148	Relationship
CONTF	95	149	Forced continuation
FIOCT1	95	149	Field or constant
CONBG1	96	150	Beginning of constant area
FC2FB1	96	150	Factor 2 from begin column
NEWSUM	98	152	New resultant length of summary field
FC2FE1	99	151-153	Factor 2 from end column
FC2TB1	9A	154	Factor 2 to begin column
FC2TE1	9D	155-157	Factor 2 to end column
TLGTH1	9E	158	Temporary length
CONEN1	A9	159-169	End of constant area
DISP21	AB	170-171	Displacement of operand 2

Figure 5-4 (Part 7 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
LNGTH1	AD	172-173	Length for move
CWADP1	AE	174	Displacement in Build control word
DISP11	B0	175-176	Displacement of operand 1
HEXNO1	B2	177-178	Hex work field
BEGER1	B4	179-180	Beginning error table
SAVER1	B6	181-182	End of error table
LNGTH2	B8	183-184	Save area for length
IWADP1	BA	185-186	Permanent displacement in control word
KEYTL	BC	187-188	Test for control field length
DSCA5@	BE	189-190	Address of #DSCA5 in O.$DSCA
SUMTB@	BE	189-190	Address of summary table
DSCA6@	C0	191-192	Address of #DSCA6 in O.$DSCA
DSCA8@	C2	193-194	Address of #DSCA8 in O.$DSCA
CONST@	C4	195-196	Address constants in O.$DSCA
CASW1	C5	197	Bit 0 Set XR1 = XR2 Bit 1 Force sequence Bit 2 Summary V found Bit 3 Displacement is -256 Bit 4 Drop key: no data or summary specifications Bit 5 SORTRS: resultant length feature Bit 6 Not SORTRS, but S card Bit 7 Too many errors
CASW2	C6	198	Not used
CCDLN1	C8	199-200	Work field to check work record length
WRECD	CA	201-202	Current displacement in work record
OFLOW@	CC	203-204	Overflow table entry
NSUM@	CE	205-206	Next summary table entry

Figure 5-4 (Part 8 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
SUM#	CF	207	Summary table element counter
SUMI	83	128-134	Summary table elment
Phase 0D Area (overlays Phase 0C equates)			
ERRORQ	83	131	00 No errors 01 Phase I main storage assigned is too small 02 Phase I main storage too small for minimum WBUFL 03 Phase II main storage too small for minimum merge 04 Phase III main storage too small for minimum WBUFL, OM=2 06 For OM=2, WRECL > WBUFL 07 For OM=4, WRECL > WBUFL 08 Phase I main storage too small for WBUFL, OM=4 09 Phase III main storage too small for minimum buffer 0A Attempted to force O.$DS1B, but not enough room
PGML1A	85	132-133	O.$DS1A active length
PGML1B	87	134-135	O.$DS1B active length
PGML1S	89	136-137	O.$DS1S active length
PGML1X	8B	138-139	O.$DS1X active length
PGML2A	8D	140-141	O.$DS2A active length
PGML3A	8F	142-143	O.$DS3A active length
FIXL1	91	144-145	Fixed Phase I length
RSAL1A	93	146-147	Remaining record storage area for Phase 1A
RSAL1S	95	148-149	Remaining record storage area for Phase 1B
RSAL1X	97	150-151	Remaining record storage area for Phase 1X
RSAL2A	99	152-153	Remaining record storage area for Phase II
MTOD#1	9B	154-155	Not used
RSAS2A	9C	156	Record storage area in sectors for Phase II
MINIL	9E	157-158	Minimum input buffer length
MINOL	A0	159-160	Minimum output buffer length
MINISL	A2	161-162	Minimum sort work area
MINISG	A4	163-164	Minimum number of records in sort area

Figure 5-4 (Part 9 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
MINISB	A6	165-166	Additional minimum sort work area
MINWL	A8	167-168	Minimum work buffer length
MIN2L	AA	169-170	Minimum Phase II (III) record storage area length
IRECLP	AC	171-172	Input record area length
WBUFLP	AE	173-174	Effective work buffer length
WBUFS	AF	175	Work buffer length in sectors
WBLKLP	B1	176-177	Effective work block length
WBLKS	B2	178	Work block length in sectors
SORTRL	B4	179-180	Internal sort record length
$OM	B5	181	Order of merge + 1
OMA	B5	181	Actual usable order of merge
OMQ	B6	182	Order of merge search switch
TOP	B9	183-185	Scratch area
$$$$	B9	185	Working storage
AVAIL	BB	186-187	Current available storage in phase
MTL1	BD	188-189	Unused storage Phase I
MTL2	BF	190-191	Unused storage Phase II
MTL3	C1	192-193	Unused storage Phase III
ADDG	C3	194-195	Additional internal sort records
ADDL	C5	196-197	Space used by ADDG
ADDOL	C7	198-199	Additional output buffer space
IBF	C9	200-201	Input blocking factor

Figure 5-4 (Part 10 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
Phases I, II, and III Area			
NXTDB@	73	114-115	Address update routine
@POP	75	116-117	Physical read/write routine
@$DS9P	77	118-119	Logical write routine
@$DS9G	79	120-121	Logical read routine
@DTAMT	7B	122-123	Data input/output
MOVER	7C	124	First byte of the fixed branch to O.$DSZA
MOVERR	7E	125-126	Last byte of the fixed branch to O.$DSZA
MOVER@	80	127-128	Move routine address
SW1X1	82	129	Bit 0 Phase II called Bit 1 Double-buffered writes in Phase II Bits 2-7 Not used
SW1X2	83	130	Not used
Disk Sort Data Management Areas			
WETBL@	84	131-132	Current work file extent table entry
OLDS@	86	133-134	Address of the old string table
I	88	135-136	String number index times 16
INQ	89	137	WGETL master switch: F = First call of pass I = Intermediate call N = Start of new string merge
OMP	8A	138	Order of merge for pass
OMS	8B	139	Order of merge for current new string
NSTR	8E	140-142	Cylinder/sector/displacement of the first block in the next old string
READ@	90	143-144	Storage address of current read
READ#	91	145	Number of sectors for current read
@AVAIL	93	146-147	Address of the high-order byte of the available list
AVAIL@	95	148-149	Address of the high-order byte of the current available element
IREC#T	98	150-152	Total number of input records

Figure 5-4 (Part 11 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
PASS	99	153	Current pass number
PASS#T	9A	154	Total number of expected passes
STR#	9C	155-156	Current string counter
STR#1	9F	157-159	Cylinder/sector/displacement of the first block of the first string
STR#T	A1	160-161	Number of strings in last pass
WBLK@	A3	162-163	Address of the current work block
WBUF@	A5	164-165	Address of the work buffer area
OBUF@	A5	164-165	Address of output buffer
WK#11	A7	166-167	Current input record length minus one
WK#22	A9	168-169	Work area
WBLKQ	AA	170	WPUTL block type: N = Start new string O = Continue old string
WCBLKC	AD	171-173	Cylinder/sector/displacement of current work block
WCSTRC	B0	174-176	Cylinder/sector/displacement of current work string
WLSTRC	B3	177-179	Cylinder/sector/displacement of last work string
WNBLKC	B6	180-182	Cylinder/sector/displacement of next work block
WORKQ	B7	183	WPUTL request switch: F = First request I = Intermediate request L = Last request
WREC@	B9	184-185	Address of the current located work record for output
WREC#	BC	186-188	Work record counter
WREC#T	BF	189-191	Total number of work records created
WRECQ	C0	192	WPUTL record position: F = Next record is first in block I = Last record is not last in block L = Last record is last in block
WRITE@	C2	193-194	Storage address of next write

Figure 5-4 (Part 12 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
WRITE#	C3	195	Number of sectors for next write
WRPBC	C5	196-197	WPUTL record and block counter
WSTRQ	C6	198	WPUTL string continuation switch: C = Continue current string E = End of current string
BUILDQ	C7	199	Work record built switch: N = No Y = Yes
Phase I Areas			
LIREC@	C9	200-201	Original address of the located input record
BREC@	CB	202-203	Address of the high-order byte of the work record build location
SREC@	CD	204-205	Address of the winning work record
SORTQ	CE	206	Master internal switch: F = First call I = Intermediate call L = Last call
SRECQ	CF	207	Winning record type: D = Dummy sort E = End sort I = Intermediate
SSTRQ	D0	208	Sort string switch: C = Continue old string N = New string
CG#	D2	209-210	Number of records in internal sort area's current string
ISORT@	D4	211-212	Address of the high-order byte of the internal sort area for Phase 1B
MVTBL	D9	213-217	Instruction to move summary table (except op code)
Free Area	E4	218-228	Not used
Phase III Areas			
WORK	C9	200-201	Work Area
AVAIL	CB	202-203	Current available storage in phase
MINOL	CD	204-205	Minimum output buffer length

Figure 5-4 (Part 13 of 14). Field Description of COMMON (Model 12 and Model 15)

Name	Hexadecimal Displacement	Bytes	Description
ERRORQ	CE	206	Error switch
ADDOL	D0	207-208	Additional output buffer space
MTL3	D2	209-210	Unused storage – Phase III
OREC#	D5	211-213	Output record counter
Free Area	E4	214-228	Not used
Job Attributes Data – Part 2			
ISTYPE	E5	229	Internal sort type: E = Exchange sort used by Phase 1A S = Selection sort used by Phase 1B with module $DS1S X = Tournament sort used by Phase 1X
ISL	E7	230-231	Length of internal sort area
ISG	E9	232-233	Number of records in internal sort area
WBF	EB	234-235	Normal work blocking factor
WBLKL	ED	236-237	Length of normal work block
WBUFL	EF	238-239	Total length of normal work buffer
OM	F0	240	Order of merge (Phase II)
OM3	F1	241	Order of merge (Phase III)
OBUFL	F3	242-243	Total length of output buffer
OBUF#	F4	244	Number of output buffers
OBLKL	F6	245-246	Length of one output block
SPLITL	F7	247	Number of tracks in a split cylinder
Constants			
I0000	F9	248-249	XL2'0000'
I0001	FB	250-251	XL2'0001'
I0002	FD	252-253	XL2'0002'
IFFFF	FF	254-255	XL2'FFFF'

Figure 5-4 (Part 14 of 14). Field Description of COMMON (Model 12 and Model 15)

The object program consists of the Select/Build routine built by Phase ØC during generation time. During execution time, Phase 1A or Phase 1B reads an input record and branches to the Select/Build routine. The Select/Build routine identifies the record and determines whether to include or omit it in the sort. If it is to be included, the Select/Build routine forms the work record from the input record; then it branches back to the calling phase (see Figure 6-5). The execution phases continue processing. See *Section 2. Method of Operation* for a control flow diagram of the execution phases and their relationship to the Select/Build routine.

The Select/Build routine is made up of three areas of code:

1. Fixed code for the linkage between the Select/Build routine and the calling phase, and between the Select/Build routine and the Pack-Include/Omit and Pack-Field routines.

2. Generated code segments selected by O.$DSCL, O.$DSCE, and O.$DSCF in Part A of Phase 0C (O.$DSCA).

3. Fixed code for the Pack-Include/Omit and Pack-Field routines.

A description of these three areas follows.

FIXED CODE FOR LINKAGE

This fixed code is always present at the beginning of the Select/Build routine. Phase 1A, 1B, or 1X enters at the address contained in BUILD@ in COMMON. The code then jumps to the generated code segments. If the record is to be included and its work record has been built, the generated code branches to the beginning of the fixed code + X'2D' (SETY). If the record is to be omitted, the generated code branches to the beginning of the fixed code + X'33' (SETN). The instructions in the fixed code set BUILDQ, a switch in COMMON, to indicate to Phase 1A, 1B, or 1X that the record is to be included or omitted. The program then returns to the next sequential instruction of Phase 1A, 1B, or 1X. The instructions which make up this fixed code follow.

```
                          0000  CCX000   EQU   *                  First byte of fixed code
0000                            SIGNW1   DS    XL1                Sign for work area 1
0001                            W1       DS    XL16               Work area 1
0011                            SIGNW2   DS    XL1                Sign for work area 2
0012                            W2       DS    XL16               Work area 2
0022  E2C5D361C2E4C9D3C4        SEL/BUILD DC   CL9 'SEL/BUILD'    Constant
002B  0004                      FOUR     DC    CL2 '04'
002D  7C E8 C7                  SETY     MVI   BUILDQ(,XR1),C'Y'  Set BUILDQ to 'Y' for an include
                                                                  record
0030  F2 87 03                           J     BLEAVE             Jump around next instruction
0033  7C D5 C7                  SETN     MVI   BUILDQ(,XR1),C'N'  Set BUILDQ to 'N' for an omit
                                                                  record
0036  C2 01 0000                BLEAVE   LA    #,XR1              Restore register 1
003A  C2 02 0000                BLEAV2   LA    #,XR2              Restore register 2
003E  C0 87 0000                BLEAV3   B     #                  Return to Phase 1A, 1B, or 1X
```

0042	34 08 004F	BENTER	ST	BLEAV3+3,ARR	Store the return address from the ARR
0046	34 01 0039	CC25	ST	BLEAVE+3,XR1	Store register 1
004A	34 02 004B	CC26	ST	BLEAV2+3,XR2	Store register 2
004E	75 02 CB		L	BREC@(,XR1),XR2	Place address of the build record in register 2
0051	F2 87 14		J	*+23	Jump over beginning of set instructions (IYES/INO)
0054	35 10 007F	VECTPK	L	PACK@,16	Load IAR with address of Pack-Include/Omit routine
0058	FFFF	CCX995	DC	AL2(PACBGN)	Entry point of Pack routine
005A	35 10 0085	PAKRNT	L	PKRNT@,16	Load IAR with address of Pack-Field routine
005E	FFFF	CCX996	DC	AL2 (CWP)	Entry point of Pack-Field routine

GENERATED CODE SEGMENTS

The generated code segments reflect the specifications on the Sequence Specifications sheet. They are placed in storage in the order in which they are generated. Figure 6-6 shows the sequence in which code segments are used forming the Select/Build routine for a given sequence specification statement. The two main functions of the generated code are to identify the record and, if the record is to be included, build the work record.

The sequence specifications for a sort are comprised of one or more sets. The two types of sets are include sets and omit sets.

Include sets identify (1) the input records to be included in the sort and (2) the fields in the records which are to be used to build the work record. Include sets are specified by one or more include statements (I in position 6) interconnected by AND and OR specifications (A or O in position 7). The last include statement is followed by one or more control field and data field statements (F in position 6). Control field specifications (N. O. or F in position 7) precede any data field specifications (D or S in position 7). The last set in the sequence specificatons must be an include set.

Omit sets identify the input records to be omitted in the sort. Omit sets are specified by one or more omit statements (O in position 6) interconnected by AND and OR specifications (A or O in position 7). Omit sets have no field specifications.

Each set is made up of subsets. A subset is the part of a set which extends:

1. From the beginning of the set up to:
   a. The first OR specificaton, or
   b. The first field specification (include set with no ORs), or
   c. The end of the set (omit set with no ORs).

2. From the beginning of the preceding subset up to:
   a. The next OR specification, or
   b. The first field specification (include set), or
   c. The end of the set (omit set).

See Figure 6-1 for an example of these requirements.

Each set begins with an IYES branch. For an include set, this branch goes to the code generated for the field specifications for this set. For an omit set, this branch goes to the beginning of the fixed code + X'33' to set BUILDQ to N, to omit this record, and return to the next sequential instruction of Phase 1 (see *Fixed Code for Linkage* in this section for more information on linkage to the calling phase).

Each subset begins with an INO branch and ends with an unconditional branch to the IYES beginning the set. INO branches to the beginning of the next set or subset. The instructions between the two branch statements are a subset consisting of the code segments for identifying an input record. Figure 6-2 shows the general structure of a set and its subsets.

	HSORTR	Header statement
Include Set	IƀC IAC	Subset meeting requirement 1a
	IOC	Subset meeting requirement 2a
	IOC IAC IAC	Subset meeting requirement 2b
	FNC FNC FDC	Field specifications
Omit Set	OƀC OAC	Subset meeting requirement 1c
Include Set	IƀC FNC	Subset meeting requirement 1b
Omit Set	OƀC OOC	Subset meeting requirement 1a Subset meeting requirement 2c
Include Set	Iƀƀƀƀƀƀƀ FNC FNC	Subset meeting requirement 1b (include-all)
	HSORTR	Header statement
Include Set	FNC FNC FDC	Subset meeting requirement 1b (implied include-all)

Figure 6-1. Example of Subset Requirements

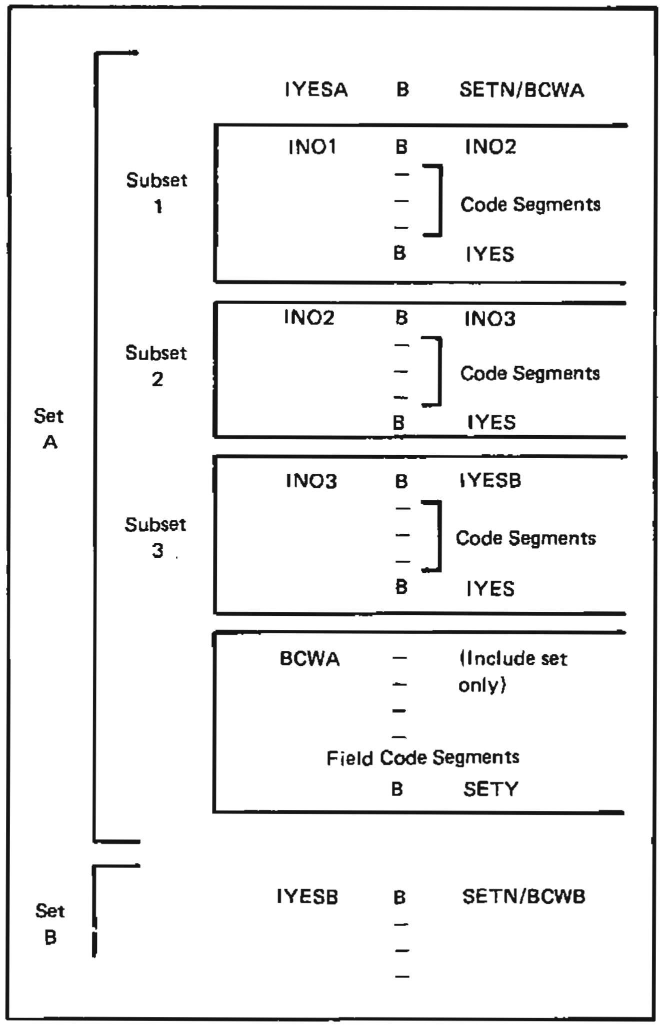


Figure 6-2. Set and Subset Structure

The last unconditional branch by a subset to IYES in an include set is followed by the code to build the work record. Generated code builds the control word of the work record. The length of the control word is specified by the longest total control field of any record type entry (positions 13-17) on the header statement. The control word is built from left to right in the order in which the control fields are specified on the Sequence Specificatons sheet (Figure 6-3). If the fields for the record are less than the total length, the control word will be padded to the right with hexadecimal zeros. Prior to calling the Select/Build routine, Phase I initializes the control word to hexadecimal zeros.

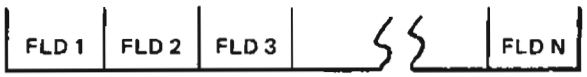


Figure 6-3. Building the Control Word

The type of job specified determines the type of information that follows the control word. For SORTA jobs, code is generated to place the 3-byte relative record number (located in IREC#T in COMMON) after the control word. For SORTR jobs, code is generated to place the data fields after the control word. The data fields are placed left to right in the order in which they are specified on the Sequence Specifications sheet. If the data field length for an include set is less than the total length, the data field will be padded to the right with X'40's. Prior to calling the Select/Build routine, Phase I initializes the data area to X'40's.

After the work record is built, the code generated for an include set ends with a branch to the beginning of the fixed code + X'2D' to set BUILDQ to Y and return to the next sequential instruction of Phase I (see *Fixed Code for Linkage* in this section for more information on linkage to the calling phase).

The last set of generated code is followed by an unconditional branch to the beginning of the fixed code + X'33' to set BUILDQ to N and return to the next sequential instruction of Phase I. Thus, *all records which are not explicitly included by meeting the requirements of any preceding generated code are omitted.*

A description of each code segment follows. The following abbreviations are used in the instructions which make up the code segments:

Abbreviation	*Meaning*
F1	To location (positions 13-16) -1 of Factor 1 (positions 9-16) or to location (positions 13-16) of control field statement
F2	To location (positions 24-27) -1 of Factor 2 (positions 20-39)
L	Length of Factor 1 (positions 9-16) or of the location (positions 9-16) field
CONSAT	Address of the Factor 2 (positions 20-39) constant
IWA	Work area = (first byte of the Select/Build routine)
CWD	Control word displacement from XR2 (XR2 contains the address of the current place in the control word)
ENFORC	One byte past force sequence of instructions
BCW	Build Control Word, the address of the code segments which build the control word

Record Identification Code Segments (Record Type Statements)

Beginning of a Set (Code Segment 1)

```
IYES    B              BCW/Fixed Code +X'33' (OMIT)
```

The code segments which identify the record type branch to this code segment if the record does meet the specifications. The branch IYES takes depends on the record type. For an omit record, IYES branches to the instructions in the fixed code which set a switch informing the program that the current record should be omitted (beginning of the fixed code + X'33'). For an include record, IYES branches to the code segments which build the control word (BCW).

Beginning of a Subset (Code Segment 1A)

```
INO     B               NEXTSET
```

The code segments which identify the record type branch to this code segment if the record does not meet the specifications. INO then branches to the first instruction following the INO branch of the next subset including the first subset of a new set (NEXTSET). The last subset is an implied omit. It branches to the instructions in the fixed code which set a switch informing the program that the current record should be omitted.

Forced Field – Zone (Part 1) (Code Segment 2)

```
        MNN     IS2021+1,F1     Set numeric equal
IS2021  CLI     F1,X'00'        Are zones equal
        BE      IS2151          Branch into MACRO 15
        CLI     F1,C'           Special zone test
```

If an F is specified in position 7 and a Z is specified in position 8, this code segment tests the zone portion of the position specified by the location field to the zone portion of the record character specified in column 17.

The zone test for a C, D, and F zone or an &, minus (-), and a blank character uses the entire code segment. For all other zone tests, only the first two instructions are used because a test for special characters is not necessary.

Zone (Code Segment 2)

```
        MNN     IS2021+1,F1     Set numeric equal
IS2021  CLI     F1,X'00'        Are zones equal
        BE      IYES/INO/*+11
        CLI     F1,C'           Special zone test
```

If a Z is specified in position 8, this code segment tests the relationship of the zone portion of the positions specified by Factor 1 to the zone portion of the constant specified.

The zone test for a C, D, and F zone or an &, minus (-), and a blank character uses the entire code segment. The branch that the BE instruction takes is determined as follows:

1. EQ relationship specified

Current Statement (col 7)	*Next Statement (col 7)*	*Branch To*
A/♭/O	A	*+11
A/♭/O	♭/O	IYES

2. NE relationship specified

Current Statement (col 7)	*Next Statement (col 7)*	*Branch To*
A/♭/O	A	INO
A	♭/O	INO
♭/O	♭/O	*+11

All other zone tests use only the first two instructions of the code segment because a test for special characters is not necessary.

Control Word Displacement (Code Segment 3)

```
        A       L(,XR1),XR2
```

At the start of the Select/Build routine, XR2 points to the control word. The control word length -1 is added to XR2.

Force Sequence – Leading Instruction (Code Segment 4)

```
        MVI     CWD(,XR2),X'FF'
```

If an F is specified in position 7 and the first force line does not have a continuation punch, this code segment is generated as a forced sequence test is entered. It moves the highest possible value into the control word build area for an ascending sequence (X'FF') and the lowest possible value for a descending sequence (X'00'). If none of the record characters are found, this default value is used in the control word.

Digit – Field to Field (Code Segment 5)

```
        ZAZ     IWA+L-1(L),F2(L)        Move Factor 2,
                                        clearing zones.
        SBN     IWA+L-1,X'F0'           Set last zone
                                        positive.
        ZAZ     IWA+L-1+L(L),F1(L)      Move Factor 1,
                                        clearing zones.
        SBN     IWA+L-1+L,X'F0'         Set last zone
                                        positive.
        CLC     IWA+L-1+L(L),IWA+L-1
```

If a D is specified in position 8 and an F is specified in position 19, this code segment tests the relationship of the digit portion of the positions specified by Factor 1 to the digit portion of the positions specified by Factor 2.

Digit – Field to Constant (Code Segment 6)

```
ZAZ     IWA+L-1(L),F1(L)     Move Factor 1,
                             clearing zones.
SBN     IWA+L-1,X'F0'        Set last zone
                             positive.
CLC     IWA+L-1(L),CONSAT
```

If a D is specified in position 8 and a C is specified in position 19, this code segment tests the relationship of the digit portion of the positions specified by Factor 1 to the digit portion of the constant specified.

Character – Field to Constant (Code Segment 7)

```
CLC     F1(L),CONSAT
```

If a C is specified in position 8 and a C is specified in position 19, this code segment tests the relationship of the characters in the positions specified by Factor 1 to the constant specified.

Character – Field to Field (Code Segment 8)

```
CLC     F1(L),F2
```

If a C is specified in position 8 and an F is specified in position 19, this code segment tests the relationship of the characters in the positions specified by Factor 1 to the characters in the positions specified by Factor 2.

Jump Over Constant (Code Segment 9)

```
J       CONSAT+1
```

CONSAT is the low-order byte of the constant. Constants are placed in the program as they are encountered in the specifications. This code segment jumps the length of the constant to continue with the rest of the program.

Normal Control or Data Field – Character or Packed (Code Segment 10)

```
MVC     CWD(L,XR2),F1
```

If an N, D, or S is specified in position 7 and a C or P is specified in position 8, this code segment moves the characters of the positions specified by the location field to the work record build area.

Summary Specification – Character or Packed (Code Segment 10A)

```
MVI     CWD(,XR2),#
MVC     CWD-1(L,XR2),CWD(,XR2)
MVC     CWD(L,XR2),F1
```

This code segment is used if: FS is specified in positions 6-7, P is specified in position 8, and a valid entry is made in positions 20, 21, and 22 of the summary specifications.

The first instruction is used to initialize a field in the work record. If Character or Packed is specified, X'00' is propagated through the field. Then the code segment moves the field specified from the input record.

Normal Control or Data Field – Zone (Code Segment 11)

```
MZZ     CWD(,XR2),F1
SBF     CWD(,XR2),X'0F'      Set numeric
                             portion off.
```

If an N, D, or S is specified in position 7, and a Z is specified in position 8, this code segment moves the zone portion of the position specified by the location field to the work record build area.

Opposite Field – Digit (Code Segment 12)

```
MVI     CWD(,XR2),X'F9'
MVC     CWD-1(L,XR2),CWD(,XR2)
MZZ     CWD(,XR2),F1
SZ      CWD(L,XR2),F1(L)
SBN     CWD(,XR2),X'F0'
```

If an O is specified in position 7 and a D is specified in position 8, this code segment moves 9's into the control word area with the same sign as the location fields. The digit portion of the information specified in the location field is then subtracted from the 9's so that the opposite digit remains in the control word build area.

Forced Field – Digit (Part 1) (Code Segment 13)

```
         MZZ       IS2131+1,F1
IS2131   CLI       F1,X'00'
```

If an F is specified in position 7 and a D is specified in position 8, this code segment compares the digit portion of the position specified by the location field to the digit portion of the record character specified in position 17.

Forced Field – Character (Part 1) (Code Segment 14)

```
         CLI       F1,X'00'      Is digit of input
                                 equal?
```

If an F is specified in position 7 and a C is specified in position 8, this code segment compares the character of the position specified by the location field to the record character specified in position 17.

Forced Field – Character, Zone, Digit (Part 2) (Code Segment 15)

```
         JNE       NEXT15        Doesn't meet
                                 test, next try.
IS2151   MVI       CWD(,XR2),X'00'  Meets test, move
                                 in substitute
                                 character.
         B         ENFORC        Leave series of
                                 force sequence
                                 tests.
NEXT15   EQU       *
```

If the contents of specified position (digit portion, zone portion, or character of a forced field) does not compare equal, a jump is taken, and the next code segment is executed. If the information does compare equal, the substitute character specified in position 18 is moved to the control word build area.

Force-All (Code Segment 16)

```
         MVI       CWD(,XR2),X'00'  Unconditional
                                 move of
                                 substitute
                                 character.
```

If a force-all line is indicated, this code segment moves the substitute characters specified in position 18 to the control word build area without any testing.

Branch to Include/Omit (Code Segment 17)

```
         B         AAX100        Branch to include
                                 return to Phase I.
```

This code segment branches to the *Fixed Code for Linkage* at either of two times:

- After the control word is built.
- Immediately after the record is identified, if no control fields are specified.

Normal Control or Data Field – Digit (Code Segment 18)

```
         ZAZ       CWD(L,XR2),F1(L)  Move Factor 1,
                                 clearing zones.
         SBN       CWD(,XR2),X'F0'  Set last zone
                                 positive.
```

If an N is specified in position 7 and a D is specified in position 8, or if a D or S is specified in position 7 and a D or U is specified in position 8, this code segment moves the digit portion of the position specified by the location field to the work record build area.

Summary Specification – Unpacked or Digit (Code Segment 18A)

```
         MVI       CWD(,XR2),X'F0'
         MVC       CWD-1(L,XR2),CWD(,XR2)
         ZAZ       CWD(L,XR2),F1(L)
         SBN       CWD(,XR2),X'F0'
```

This code segment is used if: FS is specified in positions 6-7, U or D is specified in position 8, and a valid entry is made in positions 20-22 of the summary specifications.

The first two instructions propagate X'F0' throughout the work record build area. Then the digit portion of the position specified by the location field is moved to the work record build area. The last instruction is used only for digit specifications.

Branch on Condition Instruction (Code Segment 19)

```
         BC        IYES/INO
```

This code segment follows each record identification test and is used to determine if a particular condition is met. The exact instruction depends on the record specifications. Figure 6-4 shows the resulting branch instruction and where it will branch. For example, if an O is specified in position 7 of the current specification card, and an O is specified in position 7 and an NE relationship is specified in positions 17-18 of the next specification statement a BNE to IYES is generated.

Relationship Specified (Positions 17-18)	Current Statement	b/O	b/O	A	A/b/O
	Next Statement	O	A	A/O	No more statements for this record type
EQ		BE to IYES	BNE to INO		
NE		BNE to IYES	BE to INO		
LT		BL to IYES	BNL to INO		
GT		BH to IYES	BNH to INO		
LE		BNH to IYES	BH to INO		
GE		BNL to IYES	BL to INO		

Note: The entries given for Current Statement and Next Statement refer to the entries for position 7 of the current specifications and the next specifications as follows:

b = blank
O = OR
A = AND

Figure 6-4. Table of Branch on Condition Instructions

Branch Instruction (Code Segment 19)

```
        B           IYES/INO
```

This code segment occurs at the end of a subset and is used if none of the tests in the subset are met. The branch taken is dependent on the previous BC instruction. If the BC instruction branches to IYES, the B instruction branches to INO; likewise, if the BD instruction branches to INO, the B instruction branches to IYES.

Beginning of Force Lines (Code Segment 20)

```
        J           NEXT
        B           ENFORC
NEXT    EQU         *
```

This code segment is used to leave the forced sequence tests. ENFORC is the first instruction past the current entries of forced sequence tests.

Signed Decimal Integer – PACKED (Code Segment 21)

```
        SLC     IWA+33(34),IWA+33     Clear the work area
        MVC     IWA+16(L),F1          Move Factor 1
        MVC     IWA+33(L),F2          Move Factor 2
        B       VECTPK                Branch to the Pack-Include/ Omit routine
```

This code segment is used if P is specified in position 8. This code moves the information in Factor 1 and Factor 2 to the work area. Then it branches to the Pack-Include/ Omit routine. (See *Fixed Code for Pack-Include/Omit and Pack-Field Routines.*)

Signed Decimal Integer—UNPACKED (Code Segment 22)

```
ZAZ     IWA + 16 (L), F1          Move Factor 1
ZAZ     IWA - 33 (L), F2          Move Factor 2
SZ      IWA + 16 (L), IWA + 33 (L)  Set condition register
```

This code segment is used if U is specified in position 8. This code segment sets up Factor 1 and Factor 2 in the work areas, and then performs a Subtract Zone to set the condition register.

Move Input Record (Code Segment 23)

```
        ST      IS9231+3,XR2
        MVC     IS5231(2),IIREC@(,XR1)
        J       IS7231
IS4231  DC      AL2(IREC@)
IS5231  DC      AL2(#)
        DC      AL2(IRECL)
IS7231  LA      IS4231,XR2
        B       MOVER(,XR1)
IS9231  LA      #,XR2
```

This code segment is used when data is to be read for the specified alternate collating sequence. This code segment moves the input record of data to a fixed area so the data can be used.

Opposite Field — Character, Zone (Code Segment 24)

```
MVI     CWD(,XR2),X'FF'
MVC     CWD-1(L,XR2),CWD(,XR2)
SLC     CWD(,XR2),F1
SBF     CWD(,XR2),X'0F'    Set numeric to
                           zero
```

If an O is specified in position 7 and a C or Z is specified in position 8, this code segment moves F's into the control word area. The character specified in the location field is then subtracted from the F's so that the opposite character remains in the control word build area. The last instruction in this code segment used only if a Z is specified in position 8, sets off the digit portion of the opposite character.

Overflow Indicator Field (Code Segment 25)

```
MVI     CWD(,XR2),C'#'
```

If a D or S is specified in position 7 and a V is specified in position 8, this code segment moves the character specified in the substitute character position (Posit. 18) of the current statement to the next byte in the work record.

Data Field — Unpacked Decimal (Code Segment 26)

```
ZAZ     CWD(L,XR2),F1(L)
```

If a D or S is in position 7 and a U is in position 8, this code segment zero and adds (zero fills) the indicated signed unpacked decimal number placing hexadecimal F's in the zones of each position except the units position (sign).

Unpacked Field Specifications (Code Segment 27)

```
        ZAZ     CWD(L,XR2),F1(L)
        JNL     NEXT27
        MVC     W1(L),CWD(,XR2)
        MVC     CWD(L,XR2),FFFFS
        SLC     CWD(L,XR2),W1
NEXT27  EQU     *
```

If the specification is a field specification and column 8 contains a U, the code segment shown is used for normal sequence. If opposite sequence is desired, the JNL is changed to JL.

Packed Field Specifications (Code Segment 28)

```
MVC     W1(L),F1A
B       PACKCW
DC      IL1'0'
MVC     CWD(L,XR2),W1-1
```

If the specification is a field specification and column 8 contains a P, the code segment shown is used. The B PACKCW entry passes control to the Pack-Field routine placed in storage by $DSCC.

FIXED CODE FOR PACK-INCLUDE/OMIT AND PACK-FIELD ROUTINES

The fixed code for the Pack-Include/Omit and Pack-Field routines immediately follows the generated code segments if a Pack-Include/Omit or Pack-Field numeric field has been specified on a record type statement. See *Signed Decimal Integer* (Code Segment 21 or 22) for the branch to this code.

The Pack-Include/Omit routine compares factor 1 to factor 2 and sets the program status register (PSR) before exiting. Factor 1 and Factor 2 are moved into a 17-byte field padded to the left with zeroes.

For packed data, the sign of a negative field is changed to positive. Positive fields have a 1 placed in the leftmost byte in order to force any positive field larger than any converted negative field. The two fields are then logically compared. If the result of the compare is high and Factor 1 is negative, the PSR is loaded with a low condition. If the result of the compare is low and Factor 2 is negative, the PSR is loaded with a high condition. The PRS is reversed in these cases because both factors are negative and the smaller negative number has the larger absolute value.

The instructions which make up the fixed code for the Pack-Include/Omit and Pack-Field routines follow.

Pack-Include/Omit Routine

```
PACBGN   EQU   *              This is the Pack-Include/Omit routine
         ST    CC500+3,ARR    Store return address
         TBN   IWA+16,X'0D'   Is Factor 1
         TBF   IWA+16,X'02'   * negative ('D')
         JT    CC501          Yes
         TBN   IWA+16,X'0B'   Is factor 1
         TBF   IWA+16,X'04'   * negative ('B')
         JT    CC501          Yes
         SBN   IWA+00,X'01'   Factor 1 is positive so leftmost
                              byte is set to 1

CC501    EQU   *
         SBN   IAW+16,X'0F'   Set sign to positive
         TBN   IWA+33,X'0D'   Is Factor 2
         TBF   IWA+33,X'02'   * negative ('D')
         JT    CC502          Yes
         TBN   IWA+33,X'0B'   Is Factor 1
         TBF   IWA+33,X'04'   * negative ('B')
         JT    CC502          Yes
         SBN   IWA+17,X'01'   Factor 2 is positive so leftmost
                              byte is set to 1

CC502    EQU   *
         SBN   IWA+33,X'0F'   Set sign to positive
         CLC   IWA+16(17),IWA+33  Compare Factor 1 to Factor 2
         JE    CC500          Factor 1 equals Factor 2 or PSR
                              stays equal
         JH    CC11           Factor 1 is greater than Factor 2

CC10     TBF   IWA+17,X'01'   Is Factor 2 positive
         JF    CC31           Yes so PSR stays low

CC10A    L     FOUR#,4        No so PSR loaded with high condition
         J     CC500

CC11     TBF   #+00,X'01'     Is Factor 1 positive
         JF    CC500          Yes so PSR stays high

CC31     L     10002(,XR1),4  No so PSR loaded with low condition

CC500    B     #              Return

PACEND   EQU   *
```

Pack-Field Routine

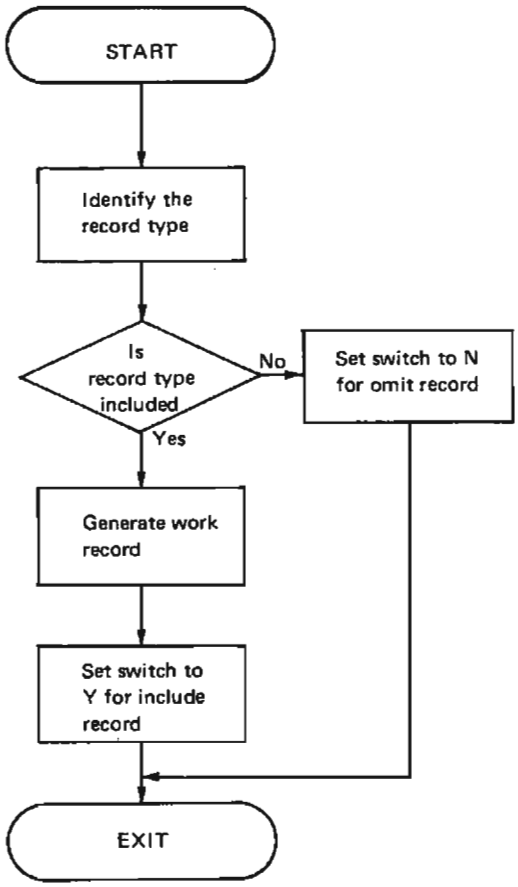
```
WORK1     EQU   16
WORK2     EQU   33
CWP       EQU   *                    This is the Pack-Field routine
RELOC1    ST    GETL+5-CWP,ARR       Store A(LEN) in GETL+5
GETL      MVC   L-CWP(1),#           Move length into 'L'
          A     I0001,(,XR1),ARR     ARR is now true ARR
RELOC2    ST    SAVE1+3-CWP,XR1      Save XR1
          LA    CWP-CWP,XR1          XR1 = relative location 0
          USING CWP,XR1
RELOC3    LA    #,XR1                XR1 = A(where pack-field loaded)
          ST    RETURN+3(,XR1),ARR   Store ARR in return
          ST    SAVE2+3 (,XR1),XR2   Save XR2
          MVC   CHK0+1(1,XR1),L(,XR1) Move LEN-1 into compare instr.
PKFLD1    SLC   WORK2(8),WORK2       Zero out WORK2 (W2)
PKFLD2    MNN   WORK2,WORK1          Move sign of W1 into W2
CHK0      CLC   WORK2(#),WORK1       W1 = 0?
          JNE   OPNORM               No, continue
PKFLD3    SBN   WORK1,X'0F'          Yes, set sign in W1 to positive
OPNORM    TBN   L(,XR1),X'80'        Opposite field specified?
          SBF   L(,XR1),X'80'        Set opposite bit off
          JF    NORMAL               Normal specified—set jumps

**        OPPOSITE FIELD SPECIFIED
          MVI   ALTER+1,(,XR1),X'80' ALTER jumps to
          MVI   NEG+2(,XR1),23       *mirror — opposite
          J     BEGIN

**        NORMAL FIELD SPECIFIED
NORMAL    MVI   ALTER+1(,XR1),X'87'  ALTER jumps to
          MVI   NEG+2(,XR1),6        * go — normal
BEGIN     MVC   SET2+3(2,XR1),OW1(,XR1) Move low-order byte @ of W1
          SLC   SET2+3(2,XR1),L(,XR1) Now SET2+3 points one byte before
          SLC   SET2+3(2,XR1),ONE(,XR1) * the number in WORK1
SET2      LA    #,XR2                XR2 = A(byte before high-order
*                                    * of WORK1 (W1) )
          MVI   0(,XR2),0            The point specified above = 0
PKFLD4    MNN   0(,XR2),WROK1        Put sign from W1 into the byte
                                     *to the left of W1
          CLI   0(,XR2),X'00'        Standard neg. number ?
          JE    NEG                  Yes, jump
          CLI   0(,XR2),X'0B'        Non-standard neg. number ?
          JNE   POS                  No, positive — jump
          MVI   0(,XR2),X'0D'        Standardize neg. number

NEG       J     #                    Mirror/go
POS       MVI   0(,XR2),X'0F'        Set units position to X'0F'
ALTER     JC    GO,#                 Mirror/go
MIRROR    MVC   WORK2(9),WORK1       W2 = W1
PKFLD6    MVC   WORK1(9),FFFS(,XR1)  W1 = X'FF'
PKFLD7    SLC   WORK1(9),WORK2       Complement
GO        SBN   WORK1,X'0F'          Set units digit to 'F'
SHIFT     ALC   WORK1(9),WORK1       Shift number to left half byte
PKFLD8    TBN   WORK1,X'08'          Shift completed ?
          BT    SHIFT(,XR1)          No, continue
          DROP  XR1
SAVE1     LA    #,XR1                Yes, restore XR1
SAVE2     LA    #,XR2                Restore XR2
RETURN    B     #                    Return to caller — $DSCF
L         DC    XL2'0000'            2-byte length so that when XR2
*                                    * is set up, it is correct
FFFS      DC    9XL1'FF'             9-byte constant of X'FF'

@W1       DC    AL2(WROK1)           @W1 = A(work area 1)
CNE       DC    XL2'0001'            2-byte constant of 1
PAKEND    EQU   *
```



To: Phase 1A
Phase 1B
Phase 1X

Figure 6-5. Select/Build Routine

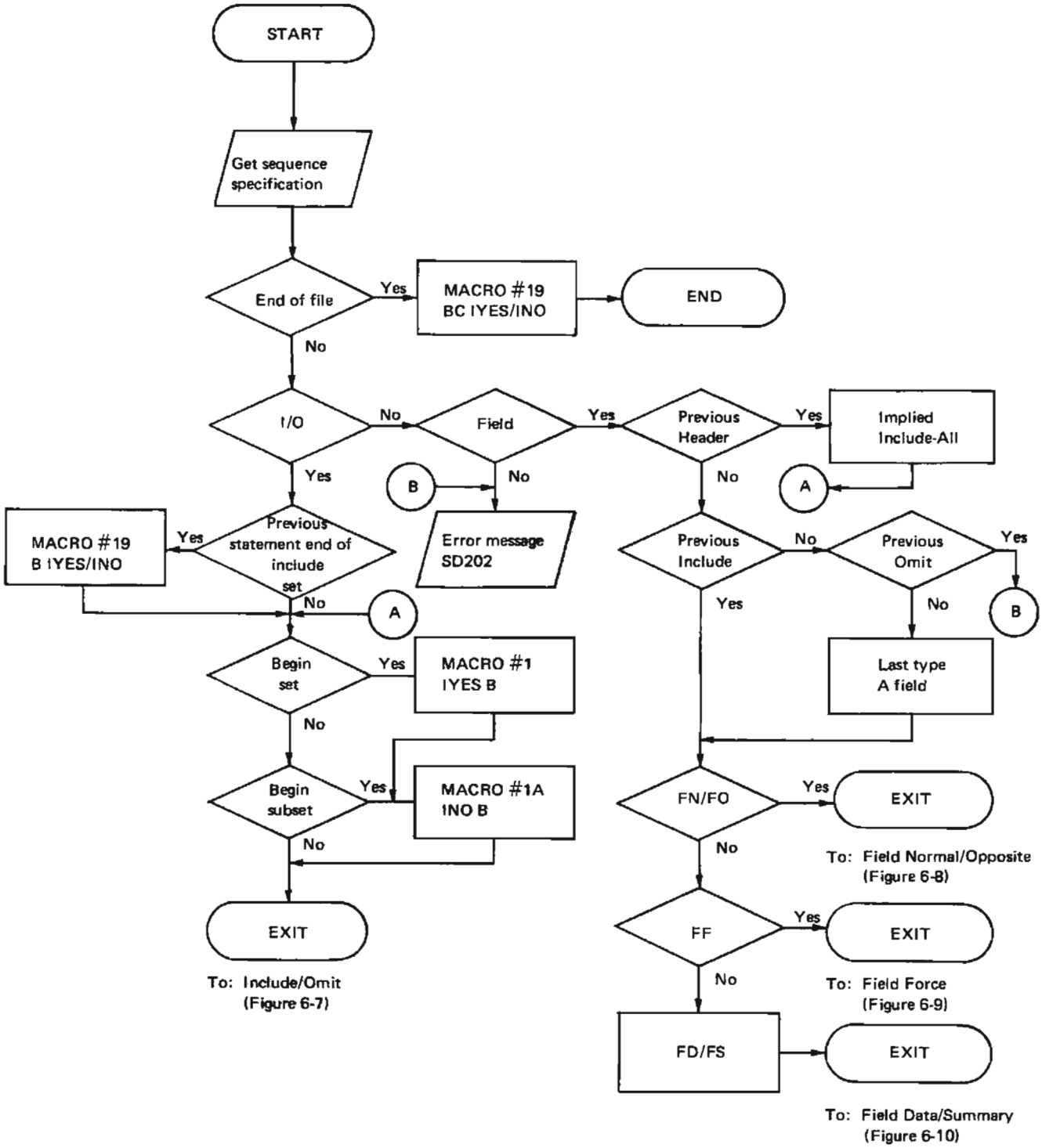
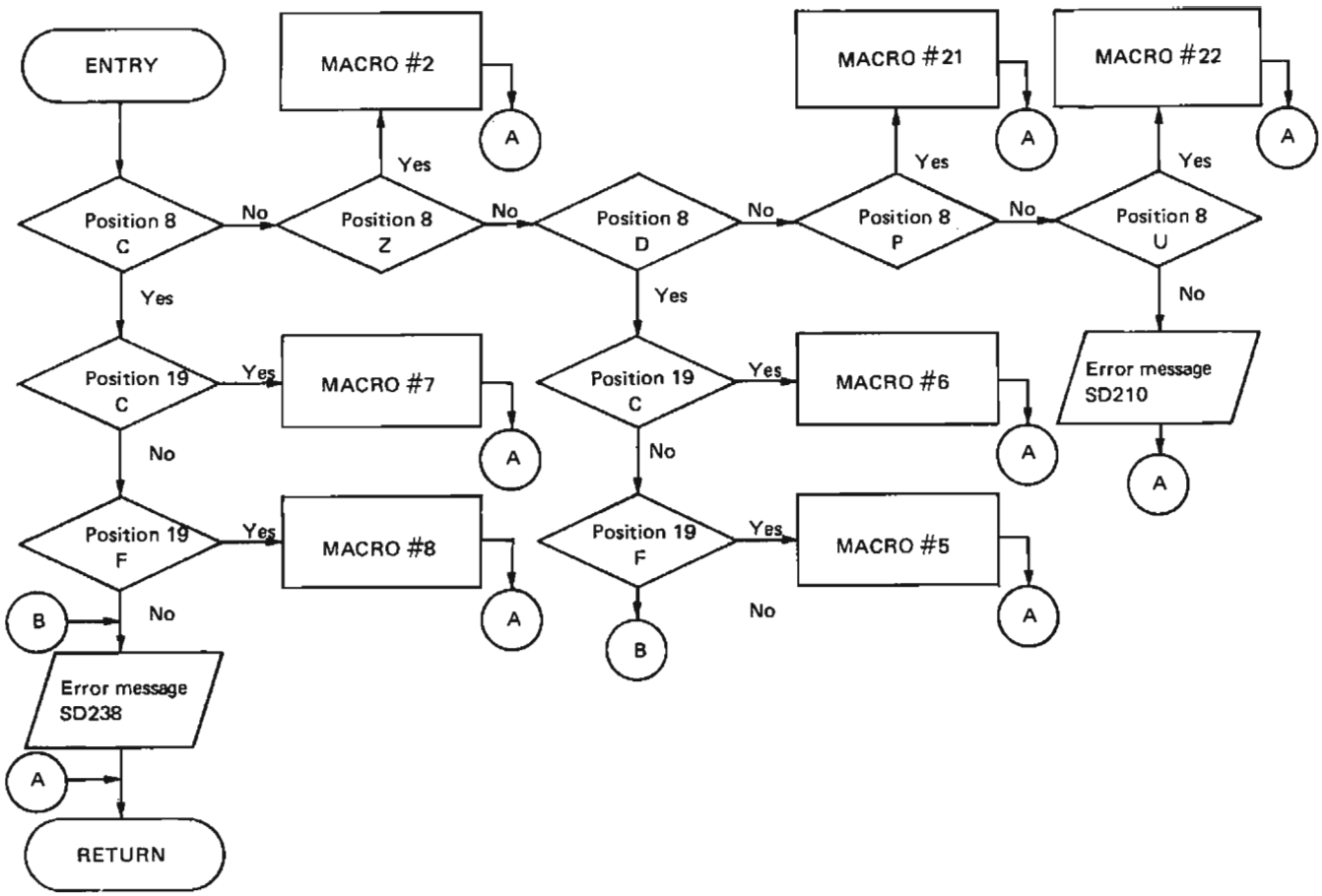


Figure 6-6. Select/Build Code Segment Selection



To: Select/Build Code Segment Selection
(Figure 6-6)

Figure 6-7. Include/Omit

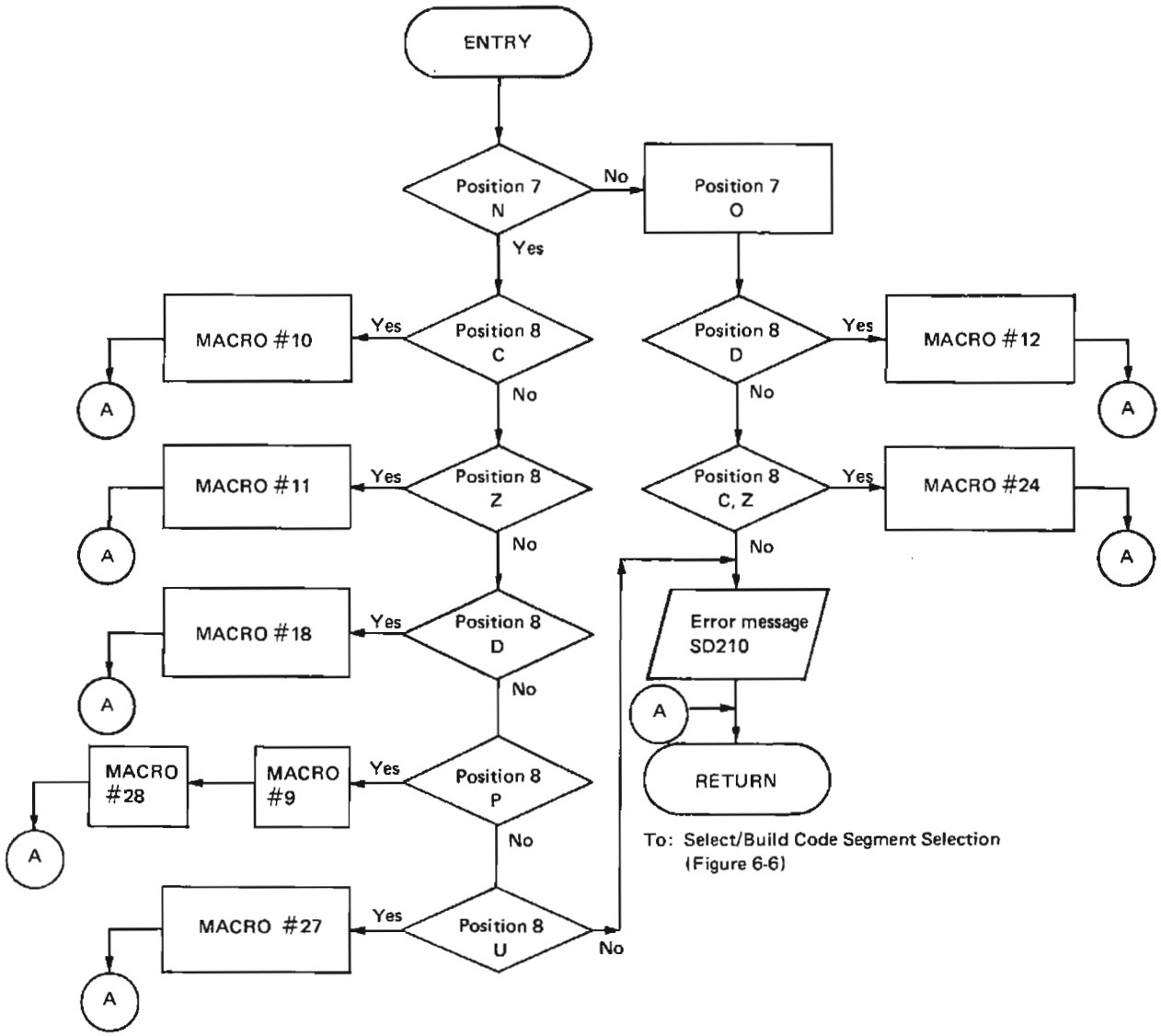


Figure 6-8. Field Normal/Opposite

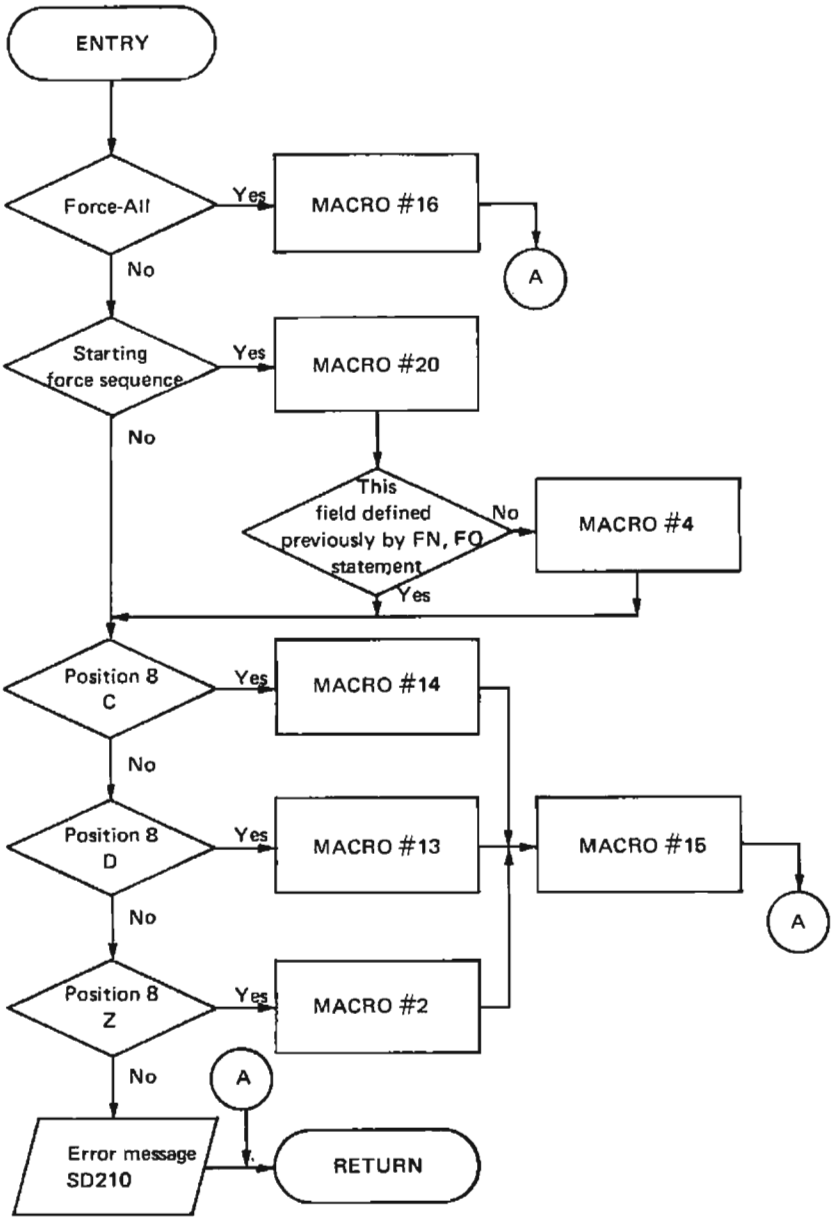


Figure 6-9. Field Force

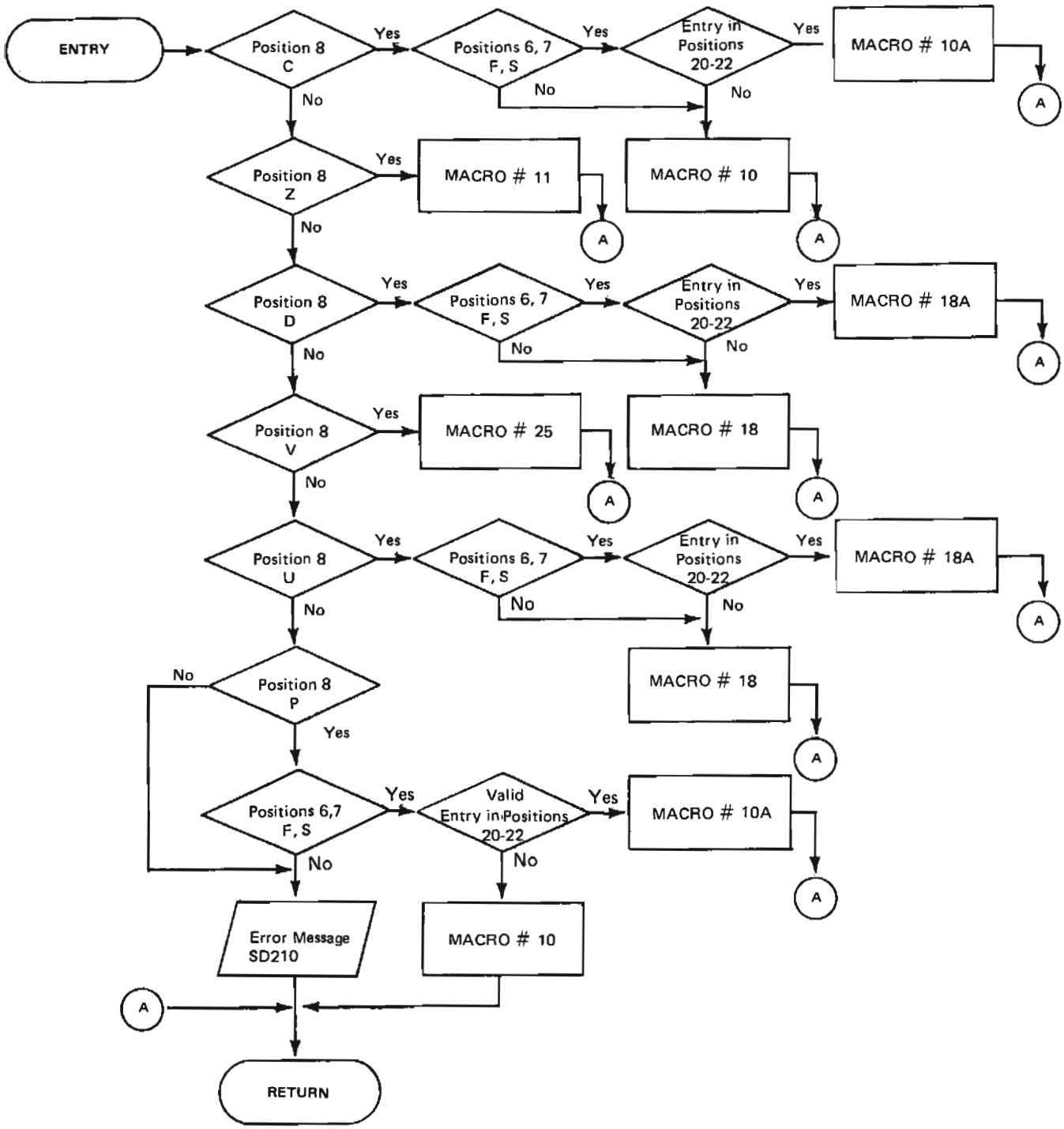


Figure 6-10. Field Data/Summary

This section describes the diagnostic aids built into the Disk Sort programs. The diagnostic aids consist of two dump routines.

DUMP ROUTINES

The first dump routine is a dynamic dump of COMMON (O.DSA routine). This routine should be used when the user is having problems with the Disk Sort program. To use O.DSA, enter DEBUG on the header statement starting in position 20. This causes O.DSE to be loaded at the end of each phase of the Disk Sort program. O.DSE determines if there is enough storage available to load O.DSA. If there is, O.DSE loads O.DSA and the dump is taken. After each dump, a program halt (of '27' on Model 10 Disk System, Model 12, and Model 15; or 'CD23' for Model 6) is displayed. At this time, a further storage dump or a disk dump may be taken. If no dumps are requested, O.DSA returns control to O.DSE which returns control to the calling routine.

The second dump routine is a terminal dump of COMMON (O.DSL routine) and is available on the Model 6 and Model 10 Disk System only. The routine can be used when a 2H program halt (on Model 10 Disk System) or 'CD25' (for Model 6) is displayed or when the job is terminated due to a processor check. In either case, record the value of the IAR, ARR, XR1, and XR2 and then take a storage dump.

Note: In a DPF environment, ensure that the value of the IAR is within the limits of the program level assigned to Disk Sort before taking a terminal dump of COMMON.

Next, press SYSTEM RESET. Then initiate O.DSL by performing a console branch to the O.$DSORT load address plus X'E5'. O.DSL is loaded into storage immediately after COMMON. O.DSL dumps COMMON and then calls the End-Of-Job Transient routine to cancel the job.

A detailed description of these routines follows.

Dynamic Dump Loader (O.DSE)

ENTRY: From: O.$DSORT, O.$DSBA, O.$DSDB, O.$DSGA, O.$DS1A, O.$DS1B, O.$DS1X, O.$DS2A, O.$DS3A, DSA or O.DSA
STORAGE MAP: See the storage map of the appropriate phase.
CHART: FA
FUNCTION:
- Computes available storage.
- Determines if there is enough available storage to load O.DSA (Disk Sort COMMON dump routine).
- Loads O.DSA if there is enough available storage.
- Prints a message and halts if there is not enough available storage.

INPUT:
- Length of calling routine.
- Length of alternate collating sequence.
- Length of O.DSA.
- Storage size for Disk Sort.

OUTPUT:
- Printed message if there is not enough storage available to load O.DSA.
- Printed title of dump if there is enough storage available to load O.DSA.

ROUTINES USED: O.DSA, O.DSD (entries for Phase ØD), O.DSX (entries for Phases I, II, and III)
EXIT:
- O.DSA if enough storage available.
- Calling routine if not enough storage available.
- Calling routine if control passed from O.DSA

Print COMMON—Dynamic Request (O.DSA)

ENTRY: From O.DSE
STORAGE MAP: See the storage map of the appropriate phase.
CHART: FB
FUNCTION: Prints Disk Sort's COMMON area with labels after Phases ØA, ØB, ØD, and all passes.
INPUT:
- Address of COMMON in XR1.
- COMTAB table (see the description of this table following this routine description).

OUTPUT: Printed output of the elements of COMMON along with their corresponding labels.
ROUTINES USED: $DSZC, $DSZE, $DSZF
EXIT: O.DSE

COMTAB Table

The COMTAB table is used by O.DSA to print the elements in COMMON. The table contains all the COMMON entries used by all phases. O.DSC contains additional COMMON entries used by Phase ØC; O.DSD

contains additional COMMON entries used by Phase ØD; O.DSX contains additional COMMON entries used by Phases I, II, and III. Each 9-byte entry in the table contains:

Byte	*Contents*
Ø-5	Character name of element in COMMON
6	Hexadecimal displacement in COMMON
7	Type of output desired
	X'ØØ' = Binary
	X'Ø1' = Decimal
	X'Ø2' = Hexadecimal
	X'FF' = Character
8	Length of element in COMMON

Print COMMON – Terminal Request (O.DSL) – Model 6 and Model 10 Disk System only

ENTRY: From user response on the console: a branch to the O.$DSORT load address + X'E5'.
CHART: FC
FUNCTION:
- Prints Disk Sort's COMMON area with labels.
- Terminates the job.

INPUT:
- Address of COMMON in XR1.
- COMTAB table (see the description of this table following the routine description of O.DSA).

OUTPUT: Printed output of the elements of COMMON along with their corresponding EBCDIC labels.
ROUTINES USED: $DSZC, $DSZE, $DSZF, O.$DS$C (entries for Phase ØC), O.DSD (entries for Phase ØD), O.DSX (entries for Phases I, II, and III)
EXIT: EOJ Transient

BITOHEX Routine ($DSZE)

ENTRY: From O.DSA or O.DSL
CHART: None
FUNCTION:
- Converts 4-bit groups into their equivalent 8-bit EBCDIC values by using a look-up table containing the EBCDIC forms of the hex digits.
- Places the 8-bit EBCDIC values into the area specified by the using routine.

INPUT:
- Address of COMMON in XR1.
- Address of a parameter list in XR2. The list contains:
  1. High-order address of the area to be converted.
  2. High-order address of the area where the EBCDIC equivalents are to be placed.
  3. Address of a 2-byte field containing the number of bytes to be converted.

OUTPUT: Converted EBCDIC values located in the area specified by the using routine.
ROUTINES USED: None
EXIT: O.DSA or O.DSL

BITOBIT Routine ($DSZF)

ENTRY: From O.DSA or O.DSL
CHART: None
FUNCTION: Converts one byte of storage to printable EBCDIC values one bit at a time.
INPUT:
- Address of COMMON in XR1.
- Address of a parameter list in XR2. The list contains:
  1. Address of the input byte.
  2. High-order address of the output area.

OUTPUT: Eight EBCDIC values in the output area specified in the parameter list.
ROUTINES USED: None
EXIT: O.DSA or O.DSL

FORCE EXECUTION PHASE OPTION

If one execution phase produces incorrect output or errors, another execution phase might correct the problems. The Force Execution Phase Option allows one execution phase to replace another.

To use this option, first check message SD600 to determine the execution phase that may have caused the problem. Next, specify a different execution phase in column 35 of the Disk Sort Header statement. The possible entries are:

Entry	*Phase*
A	1A (O.$DS1A)
B	1B (O.$DS1B)
X	1X (O.$DS1X)

If the Force Execution Phase Option can not force the execution phase, error message SD391 is logged and another execution phase should be specified.

PHASE II Merge Configuration–SD694 (Model 12 and Model 15)

On the Model 15 message SD694;

"SD694 – XX PASSES REMAINING $N_1N_2N_3$"

N_1= Order of merge to be used in Phase II.

N_2= Number of output buffers to be used during Phase II.

N_3= Order of merge to be used in Phase III.

Should you experience problems with a particular configuration, you may be able to alter the configuration selected by forcing a different Execution Phase algorithm (refer to *Force Execution Phase Option* this section), or by varying the partition size for the Disk Sort.

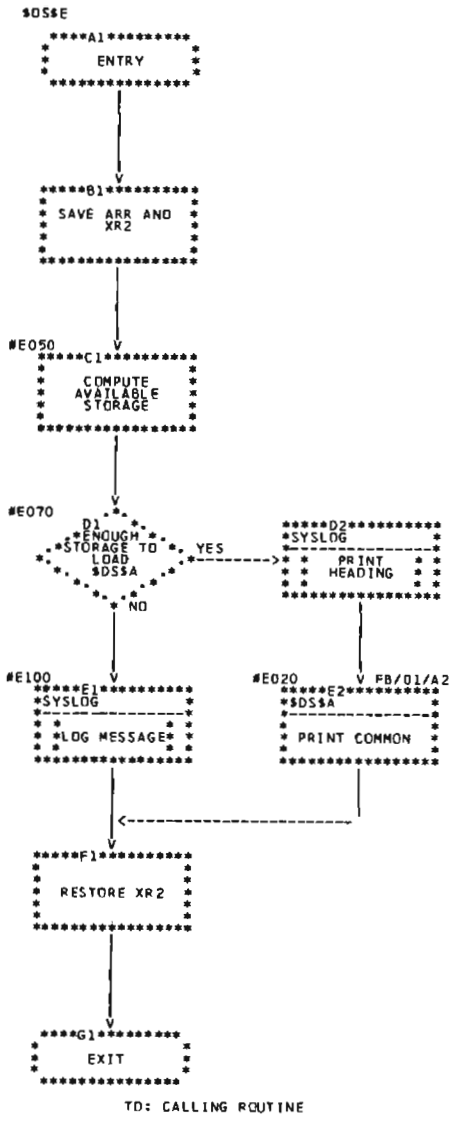


Chart FA. Dynamic Dump Loader

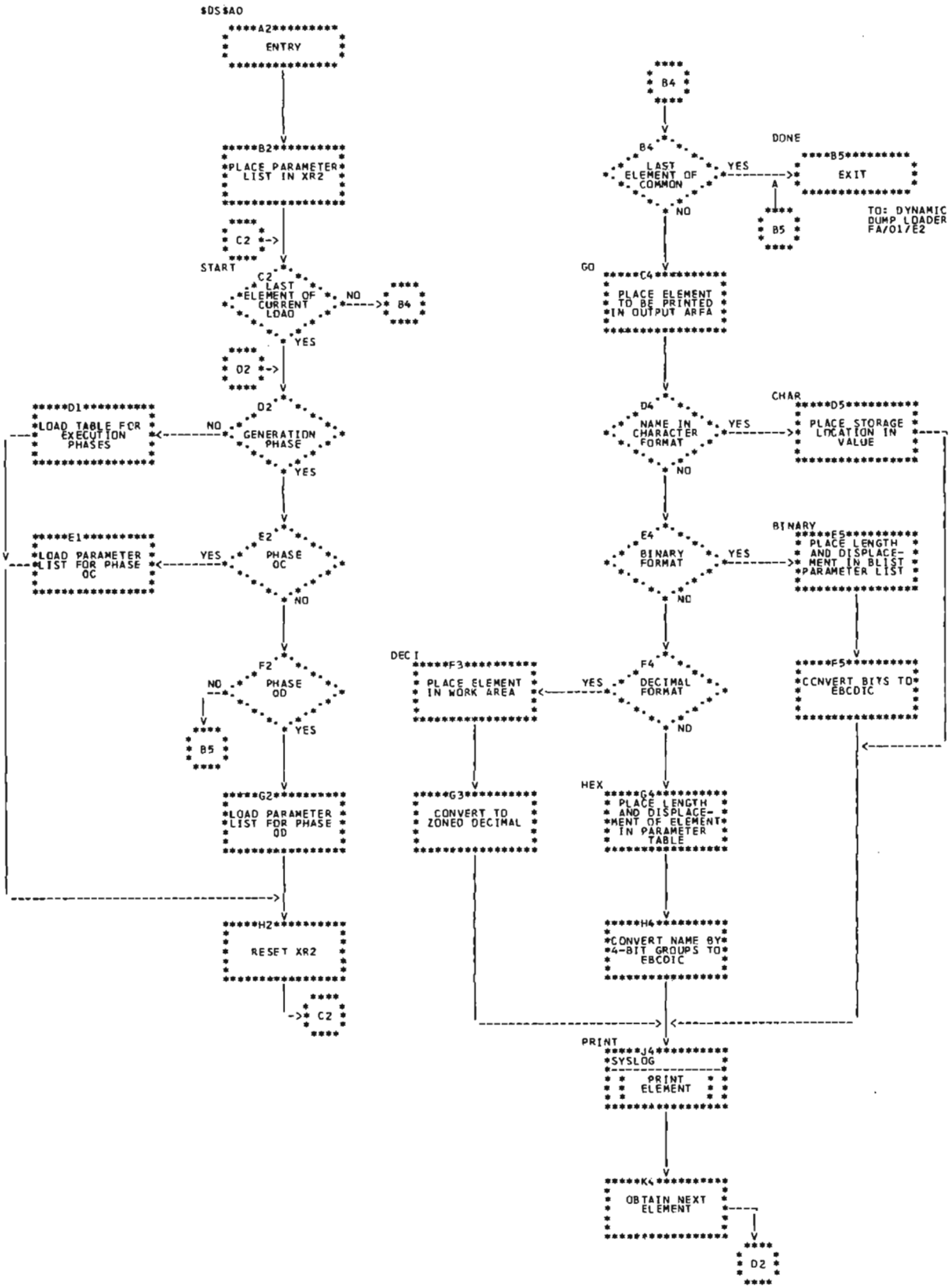


Chart FB. Print COMMON–Dynamic Request

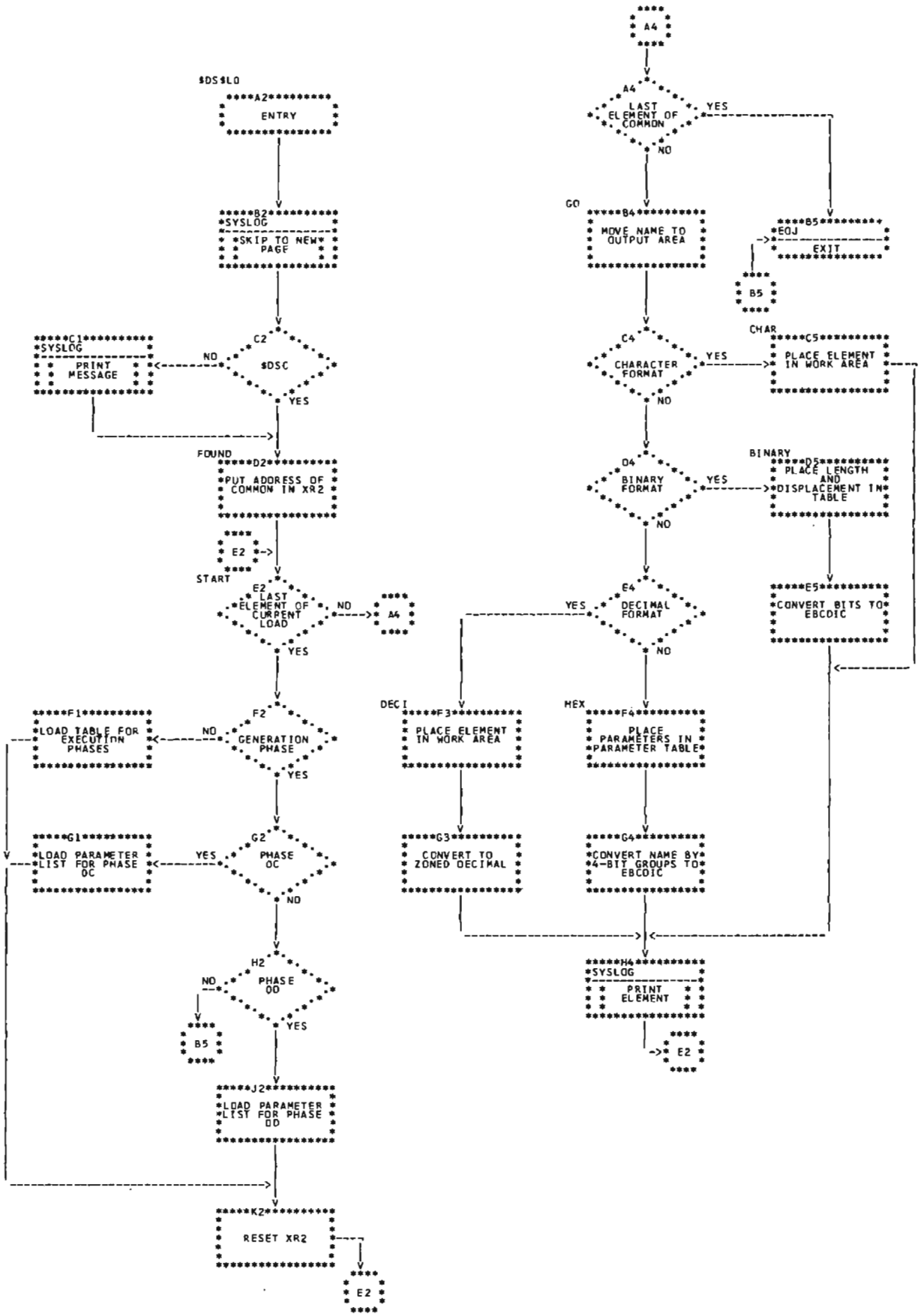


Chart FC. Print COMMON–Terminal Request

CHART NUMBERING

Flowcharts are identified in this manual in the following manner:

- A unique pair of letters identify a flowchart consisting of a single page.

  Example: AA, AB, AC

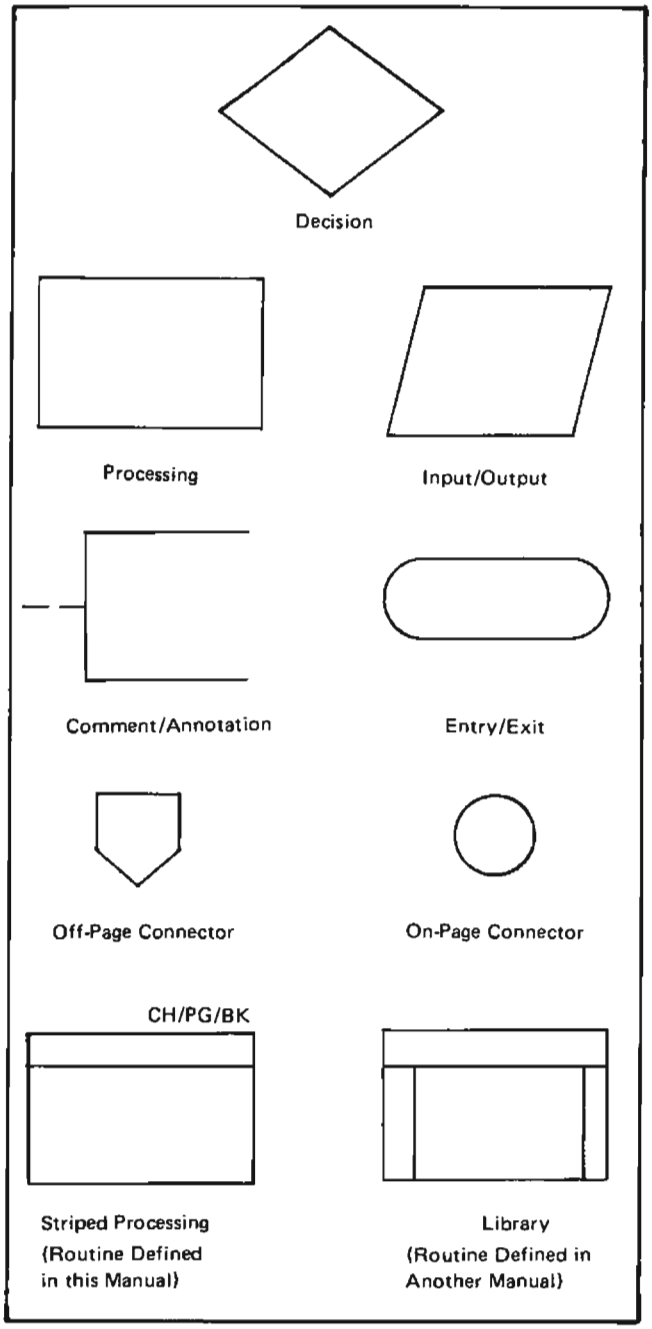
- If a flowchart consists of multiple pages, the same pair of letters identify each page, but each page has a unique number.

  Example: First page, CA-01
  Second page, CA-02
  Third page, CA-03, etc.

- Each part is assigned two sets of flowchart identifying letters. Only after the first set is completely used, that is, AA-AZ, is the second set used.

SYMBOLS

The flowchart symbols used in the program logic manual are:

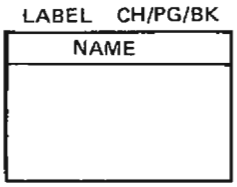


The processing, decision, input/output, and comment/annotation blocks are self-explanatory. The other blocks need further explanation:

Striped Processing Blocks

The striped processing block indicated entry to a module or routine which is flowcharted and/or described in this logic manual.

Example:



CH/PG/BK indicates the flowchart, page, and block identification where the module or routine is flowcharted. If it is not flowcharted, see the index for the location of the description of that routine.

Library Blocks

A library block indicates a function or module is documented in another manual.

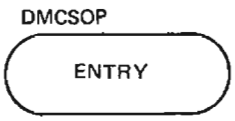
Example:

The NAME of the function/module is listed in the *Preface* of this manual by the name of the manual that contains the description of this function/module.

Entry Block

The label in the upper left corner, just above the symbol, is the entry point in the listing for that part of the program.

Example:



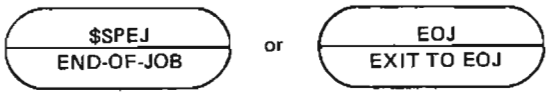
Exit Block

This block indicates that control is leaving this chart.

Example:

If control is being passed to a known function/module, which is documented in another manual, a striped exit block is used. The manual can be found via the *Preface* as with library blocks.

Example:



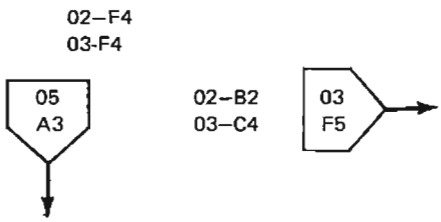
Connectors

Off-page connectors reference between different pages of the same chart ID. Off-page connectors leaving a page contain the page number and block number of their destination.

Example:

Off-page connectors contain the page and block number of their origin. If the entry point referenced by the off-page connectors is referenced from more than one origin, all origins are given.

Example:



On-page connectors contain the location of a block on the same page. On-page connectors always contain the location of the destination block.

IBM System/3 Disk Sort PLM (File No. S3-33) Printed in U.S.A. LY21-0517-6

International Business Machines Corporation
General Systems Division
5775D Glenridge Drive N.E.
Atlanta, Georgia 30301
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)