LY33-8042-6
File No. S370-33 (OS/VS)

Program Product

OS/VS Sort/Merge Logic

Program Number 5740-SM1

Release 5

IBM

Seventh Edition (September, 1979)

This edition, as amended by technical newsletters LN20-9345 and LN20-9390, applies to Release 5, Modification 0 of OS/VS Sort/Merge, Program Product 5740-SM1, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Technical newsletter LN20-9329 is obsolete.

The changes for this edition are summarized under "Summary of Amendments." Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

SUMMARY OF AMENDMENTS
FOR LY33-8042-6
5740-SM1

RELEASE 5, MODIFICATION 0

- Improved performance when sorting variable-length records with new disk sorting technique (VLR-Blockset).
- Ability to add to or change installed or passed user options, using the new OPTION control statement.
- Support of the IBM 3375 direct access storage device for initial input, final output, and intermediate work data sets.
- Ability to produce statistical data about sort applications executed via the System Management Facility (SMF).
- Ability to specify that format CH be translated the same as format AQ.
- Ability to specify that record counters should or should not be checked at the end of execution of sorting applications that use the E35 exit routine without a SORTOUT data set.

SUMMARY OF AMENDMENTS
FOR LY33-8042-6
5740-SM1

RELEASE 4

- A further standard disk sorting technique (Blockset) added to improve performance
- The design point is changed from 32K bytes to 48K bytes
- SORTIN/SORTOUT I/O handling is enhanced to improve performance.
- The default printing of the Sort specially formatted dump is removed.
- Support of the IBM 3380 direct access storage device for initial input, final output, amd intermediate work data sets.

SUMMARY OF AMENDMENTS
FOR LY33-8042-5
5740-SM1

RELEASE 3

NEW DISK SORTING TECHNIQUE

A new technique, Vale, has been introduced. Its primary function is to extend the benefits of the Peerage technique to variable-length records.

WORK DATA SETS

- The sort program's work data sets can be on a mixture of any of the supported disk types.
- If necessary, a secondary storage allocation is automatically made; this need not be specified in JCL.
- Unused space is automatically released; this need not be specified in JCL.
- Work I/O handling has been changed to give improved performance.

LONGER INPUT RECORDS FOR DISK SORT

Input records can be up to 32,760 bytes long, regardless of whether work data sets are on tape or disk.

Preface

This publication describes the functions and internal logic of the OS/VS Sort/Merge Program Product, Program Number 5740-SM1. Its purpose is to aid IBM Program Service Representatives to understand the program and to locate and fix program faults. The reader should be familiar with the information contained in related publications listed below.

This publication has six major parts:

Section 1. Introduction describes the program's structure and operational and physical characteristics and relates it to the operating system.

Section 2. Method of Operation contains method-of-operation diagrams and extended descriptions which detail the program's functions and relate them to specific code modules.

Section 3. Program Organization gives the physical organization of the program in main storage, as well as describing the interface between modules, and record movement.

Section 4. Directories serves as a cross-reference between parts of the program and their description in the manual.

Section 5. Data Areas gives the layout of important data areas used by the program.

Section 6. Debugging Aids gives diagnostic and debugging hints under twelve headings, including how to decide whether a problem is due to a program error, how to add a temporary change to a program module, the origin of error messages, and how to effect simple bypasses.

This publication also contains four appendixes:

Appendix A. Program Exits
Appendix B. Format Codes (Conventional Techniques)
Appendix C. Checkpoint/Restart Facility
Appendix D. Program Listing Standards and Conventions

RELATED PUBLICATIONS

OS/VS Sort/Merge Programmer's Guide, SC33-4035

| OS/VS Sort/Merge Installation Guide, SC33-4034

Contents

Figures

Section 1. Introduction

The OS/VS Sort/Merge Program Product has three standard components:

- Disk sort, which is reenterable
- Tape sort
- Merge only

In addition it contains two conventional disk sorting techniques, normally not used.

The total program is a generalized sort/merge program working as a processing program under the operating system. It can sort one VSAM or QSAM data set, and can merge up to 16 VSAM or QSAM data sets. Each of the QSAM data sets may consist of a number of concatenated data sets, up to the system maximum.

Input and output may be fixed or variable length, blocked or unblocked, and may be on any device supported by QSAM or VSAM. Output may be in VSAM form even if input is QSAM, as long as the output data set has previously been defined using the AMS utility.

Most sorting applications need intermediate (work) storage, which may be on:

- 2400 or 3400 series tapes
- 2314 or 2319 direct access devices
- 3330 series direct access devices (3330/3333, Model 1 and/or Model 11)
- 3340 direct access devices
- 3350 direct access devices
- 3375 direct access devices
- 3380 direct access devices
- 3380 direct access devices and 3880 Models 2 or 3 with Speed Matching Buffer
- 3380 direct access devices and 3880 Model 13 with Cache
- 3850 MSS volumes

Direct access device types can be mixed.

Relationship to the Operating System

The program communicates through macro instructions and interruptions with the operating system control program.

For example, for input and output the data management EXCP, QSAM, BSAM, or VSAM routines can be used; the EXCP (or EXCPVR, SVS, and VS1 only) macro instruction is used for intermediate storage I/O operations, and sometimes for input and output (EXCPVR is never used for input or output); and the linkage editor may be called to link-edit user-written routines to the program.

The program is designed to make use of the following facilities if they are included at system generation: Input/Output Recovery Management Support; Multiple Console Support; and (at the user's request) Checkpoint/Restart.

Program Structure

The program operates in four major phases, as indicated in Figure 1.

The first part of Phase 0 will be covered here. This part reads and interprets program control statements, and selects the technique to be used.

For a merge, there is only one technique.

TAPE SORT TECHNIQUES

If tape is specified for work areas, one of the conventional tape sorts (BALN, POLY, or OSCL) is used. The choice of technique depends solely on the numbers of work and input tapes specified.

DISK SORT TECHNIQUES

There are four standard disk sorting techniques available to the sort/merge program:

- FLR-Blockset--fixed-length records
- VLR-Blockset--variable-length records
- Peerage--fixed-length records
- Vale--both fixed- and variable-length records

Sort/merge will select one of the Blockset techniques if all the conditions for its use are met (see "Conditions for Use of Blockset Sorting Techniques").

Disk Sorting Techniques For Fixed-Length Records

The sort/merge program's most efficient fixed-length record technique, FLR-Blockset, will be used for most sorting applications, providing the conditions listed in "Conditions for Use of Blockset Techniques" are met. If one or more of the conditions for the FLR-Blockset technique are not met, the Peerage or Vale technique will be used, where possible.

Disk Sorting Techniques For Variable-Length Records

The high-performance VLR-Blockset technique will be used for sorting variable-length records if all the requirements listed in the following section are fulfilled. If not, the current variable-length disk sorting technique, Vale, will generally be used.

To enable sort/merge to attempt to select the best technique, whether VLR-Blockset or Vale, the following guidelines may be useful. If the average length of variable records is more than 350 bytes, you should specify the L5 operand on the RECORD control statement. If you specify an L5 operand that is between 350 and 1,000 bytes, Sort uses the Vale technique when the ratio of region size to number of records is large. When L5 is greater than 1,000 bytes, Vale is generally used. If the

working storage is less than 100K bytes, sort/merge will attempt to select VLR-Blockset regardless of average record length. If you don't specify L5, sort/merge will try to use VLR-Blockset.

When used, the new VLR-Blockset technique will generally show processing time improvement over Vale.

CONDITIONS FOR USE OF BLOCKSET SORTING TECHNIQUES

The sort/merge program has two high-performance disk sorting techniques, FLR-Blockset and VLR-Blockset, for fixed- and variable-length records, respectively. The program will first attempt to use one of these techniques, providing the following conditions are fulfilled. If they are not, then one of the other standard disk sorting techniques, Peerage or Vale, may be used where possible (Peerage or Vale for fixed-length records; Vale for variable-length records).

The first list below includes conditions common to both techniques. The second list includes conditions relevant to FLR-Blockset only, and the third, to VLR-Blockset only.

Conditions Common To Both Blockset Techniques

- More than about 64K bytes of main storage plus additional storage for buffers are available for sort and other possible modules in the region/partition. The larger the input/output block sizes are, the larger main storage must be.
- No program exits other than E15 and/or E35 (without overlay structures) provided they are pre-linkedited.
- If a SORTCNTL DD statement is used, no control statements other than OPTION should be included.
- Tape work data set is not specified.
- Under MVS, up to 26 dynamically allocated sort work data sets may be used, depending on the complexity of the control field and use of SMF.
- Input or output is not a VSAM or an ASCII data set, or track overflow record format (RECFM=FT).
- Input is not a direct-access data set with key sequenced organization (BDAM).
- Input or output must not be a spool or a dummy data set.
- Output cannot be padded or truncated records, or an old data set residing on tape.
- Multivolume disk data output is not requested.
- Checkpoint is not specified.
- Control fields that do not exceed 248 bytes.
- Control fields that do not cause the intermediate record to expand by more than 30% of the total record length. Factors that might expand the record are overlapping fields, decimal fields, fields that require translation, or specification of EQUALS.
- All supported control field formats except those with leading, trailing, overpunched, or separate signs, or ASCII format control fields.
- Skipping of input records is not requested.

FLR-Blockset Conditions

- SORTIN record length plus 13 bytes and any additional bytes caused by control field expansion must not exceed the smallest SORTWK track capacity or 32K bytes, whichever is smaller.
- Record length is not to be changed by program exit E15 and/or E35.
- SORTWK data sets must be allocated in cylinders (MVS only).

VLR-Blockset Conditions

VLR-Blockset minimum storage requirements are defined by the following computations (whichever results in the larger value should be used, but in no case should less than 69K bytes be used):

- In computing the amount of storage necessary to execute VLR-Blockset, use whichever of the following computations resulting in the largest value:

  1. 48K bytes of main storage plus the largest of three times:

     a. the maximum input block size, or

     b. the maximum output block size, or

     c. 2000 bytes.

  2. 48K of main storage plus four times the size of the maximum record length, plus the largest of the following:

     a. the maximum input block size, or

     b. the maximum output block size, or

     c. 2000 bytes.

- Maximum record length does not exceed the track length for the SORTIN or SORTOUT disk data set, or 32,000 bytes, whichever is smaller.
- Input or output is not spanned, variable-length records.
- Input or output is not Format D records (variable-length ASCII tape records).
- Work data sets are specified (a sort in main storage is not supported).
- The sort/merge program is not dynamically invoked by IMS/VS for variable-length record sorting applications.
- The control field does not overlap the record descriptor word (RDW).
- If the ratio of region size to the number of input records is large, and if the L5 operand specified on the RECORD control statement is greater than 350 bytes, sort/merge may, in some cases, choose to use the Vale technique. If L5 is not specified, sort/merge will execute VLR-Blockset if all other conditions are met.

Program Structure

As shown in Figure 1, the program operates in four major phases. The first part of Phase 0 reads and interprets program control statements, and selects the technique to be used.

For a merge there is only one technique.

TAPE SORT TECHNIQUES

If tape is specified for work areas one of the conventional tape sorts (BALN, POLY, or OSCL) is used. The choice of technique depends solely on the numbers of work and input tapes specified.

DISK SORT TECHNIQUES

For a disk sort the program selects one of three standard techniques: BLOCKSET, PEERAGE, or VALE.

The program's most efficient technique, BLOCKSET will be used for most sorting applications fulfilling the following conditions:

- Fixed-length records up to the length of the smallest SORTWK disk track size
- More than about 60K bytes of main storage available for sorting and possible modules in the region/partition
- No program exits other than E15 and/or E35, provided they are prelink-edited
- Input or output is not:
  - VSAM data set
  - ASCII data set
  - Data set with track overflow record format (RECFM=FT)
  - Direct access keyed sequenced data set
  - Residing on a device type not supported as a sort work area
  - Spool data set or dummy data set

If these conditions are not met, the program will attempt to use Peerage or Vale. Peerage is normally used if the following criteria are met:

- Fixed-length records
- Record length no greater than track length
- No exits to be activated other than E15, E18, E35, E39, or E61
- Control word not too long (see note)

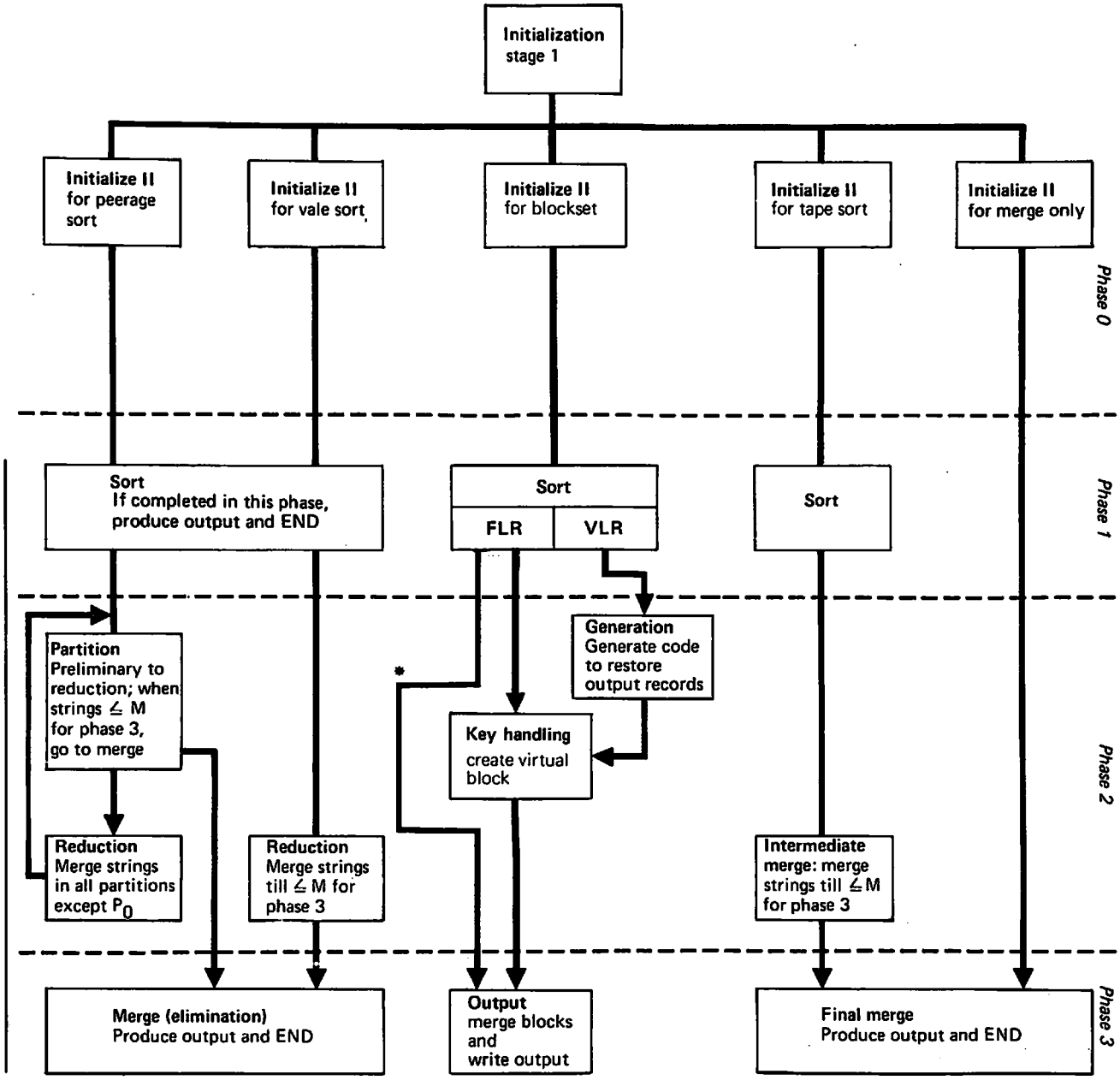
Note: No figure can be given for how long the control word can be if the Peerage technique is to be used; it depends on many variables, such as device type for work storage and amount of main storage available for buffers. However, the length limit is unlikely to be reached before 256 bytes, and will usually be considerably higher.

If any one of the conditions mentioned above is not satisfied, the program will attempt to use Vale.

The user should not normally need to be aware that these different techniques exist. In the OS/VS Sort/Merge Programmer's Guide, they are collectively referred to as the "standard disk technique." However, it is possible to specify, when the program is installed or in the OPTION or DEBUG statement on execution, that a Blockset technique should not be used.

An informational message (ICE092I or ICE093I) states which of the standard disk techniques has been used.

The conventional disk sorts supplied with the program (BALN and CRCX) can be forced by a parameter of the DEBUG statement. Care should then be taken that the SORTWK requirements for the forced technique have been met. For information on these requirements, refer to the OS/VS Sort/Merge Programmer's Guide.



*If all records fit in main storage or just 1 string of data written on SORTWKs, the key handling is bypassed.

Figure 1. Phases of the Program

Operational Considerations

The program may be initiated by an EXEC statement or invoked from a processing program, as shown in Figure 2.

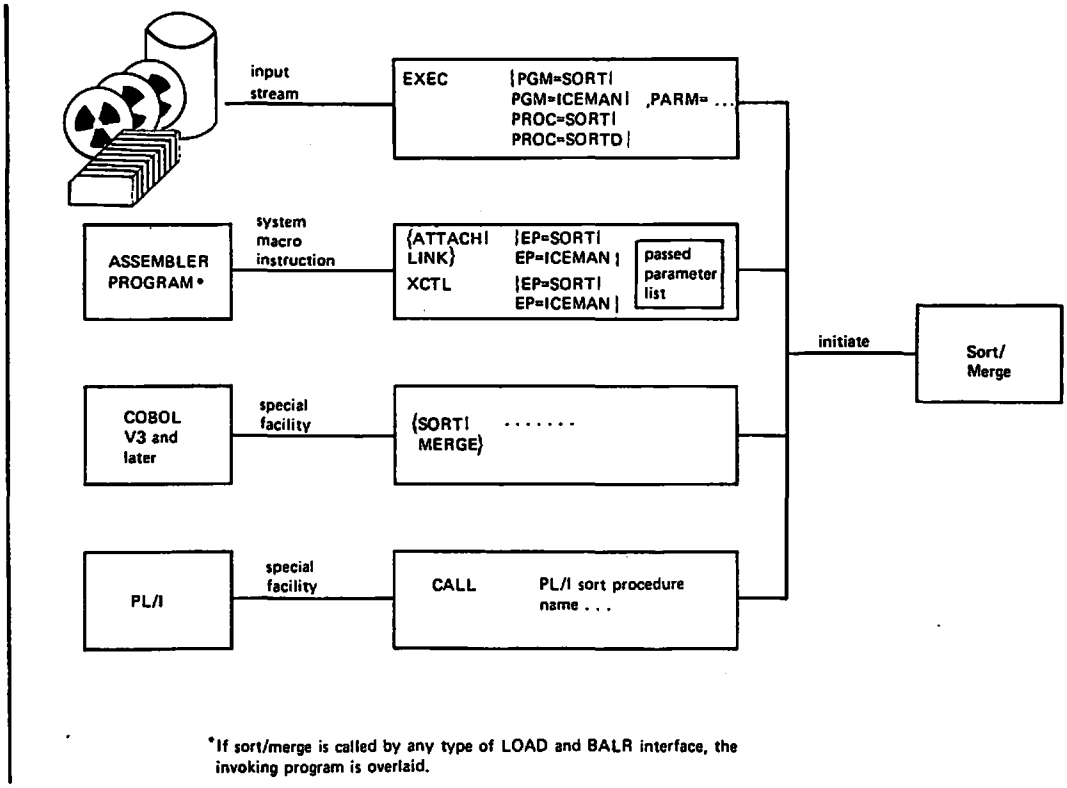


Figure 2. Initiation of the Program

For an EXEC sort or merge, control information is supplied via the following control statements in SYSIN: SORT, MERGE, OPTION, RECORD, MODS, ALTSEQ, DEBUG, and END.

For a sort or merge invoked from an assembler-language program, control information is supplied in a parameter list and SORTCNTL. For COBOL and PL/I, additional information may be passed via the SORTCNTL data set.

Figure 3 gives an example of the use of the ATTACH macro instruction.

User routines for use at program exits may be located in partitioned data sets defined by the user in the MODS statement, or may be in SYSIN, in which case they are copied into the SORTMODS data set. The user defines link-editing requirements: none, separate, or together with other routines for the same phase.

```
         LA    1,PARLST                LOAD ADDR OF PARAM POINTER IN R1
         ATTACH EP=SORT                INVOKE SORT
         .
         .
         .
PARLST   DC    X'80',AL3(ADLST)        POINTER FLAG/ADDRESS OF PARAM LIST
         .
         .
         .
         CNOP  2,4                     ALIGN TO CORRECT BOUNDARY
ADLST    DC    AL2(LISTEND-LISTBEG)    PARAM LIST LENGTH
LISTBEG  DC    A(SORTA)                BEGINNING ADDRESS OF SORT STMT
         DC    A(SORTZ)                END ADDRESS OF SORT STMT
         DC    A(RECA)                 BEGINNING ADDR OF RECORD STMT
         DC    A(RECZ)                 END ADDR OF RECORD STMT
         DC    A(MOD1)                 ADDR OF E15 RTN
         DC    A(MOD2)                 ADDR OF E35 RTN
         DC    C'ABC#'                 DDNAME CHARACTERS
         DC    X'00'                   OPTIONAL MAIN STORAGE FLAG BYTE
         DC    X'01A000'               OPTIONAL MAIN STORAGE IN HEX
         DC    X'FF'                   MESSAGE OPTION FLAG BYTE
         DC    C'(U)'                  MESSAGE OPTION
LISTEND  EQU   *
SORTA    DC    C' SORT FIELDS=(10,15,CH,A),'    SORT CONTROL STMT
         DC    C'FILSZ=4780'            (CONTINUED)
SORTZ    DC    C' '                     DELIMITER
RECA     DC    C' RECORD LENGTH=100,TYPE=F' RECORD CONTROL STMT
RECZ     DC    C' '                     DELIMITER
MOD1     DS    0H                       (routine for exit E15)
         USING *,15
         .
         .
MOD2     DS    0H                       (routine for exit E35)
         USING *,15
         .
         .
```

Figure 3. Initiation Via the ATTACH Macro

Physical Characteristics

When a standard disk technique is used, the program is reenterable.

MAIN STORAGE

The program needs a minimum of 48K bytes of main storage. If performance is a consideration you should allocate at least 72K bytes, and preferably between 128K bytes and 512K bytes.

The program will make use of all available main storage in the partition or region if SIZE(MAX) is in effect; this can be specified either at installation or at execution. The MAXLIM parameter should have been specified at installation time: this sets an upper limit to the amount of space which the program can use. If it has not been specified, and SIZE(MAX) is used in a very large virtual partition or region, performance will usually deteriorate and deactivation may result.

LIBRARIES

There are two link libraries designated for your sort/merge program--one containing the reenterable routines, and the other containing non-reenterable initiating routines. In addition, a sort library containing all other sort/merge routines as well as space in the system SVC library is designated when you are using any sort application with tape work areas or any merge application, or if any of the nonstandard disk techniques are forced. If sort/merge cataloged procedures are used, space in a procedure library is also designated.

System Libraries

The system libraries usually used are shown in Figure 4. In addition, Figure 5 shows how the data sets are used during execution of sort/merge.

System Libraries	VS1 (also MVT, MFT)	SVS, MVS
Link library (reenterable modules)	SYS1.LINKLIB	SYS1.LPALIB
Link library (non-reenterable modules)	SYS1.LINKLIB	SYS1.LINKLIB
Sort library	SYS1.SORTLIB	SYS1.SORTLIB
SVC library	SYS1.SVCLIB	SYS1.LPALIB
Procedure library	SYS1.PROCLIB	SYS1.PROCLIB

Figure 4. System Libraries

If SYS1.PROCLIB is used for procedures, any earlier SORT or SORTD procedures (SM-023 or 5734-SM1) should be scratched.

Private Libraries

If you have more than one operating system and use private libraries, only one set of modules is needed (except for the SVC routine, which is dependent on the system under which you are running; it must always be in a system library).

DATA SETS

Temporary data sets are used at execution time. These are illustrated in Figure 5.

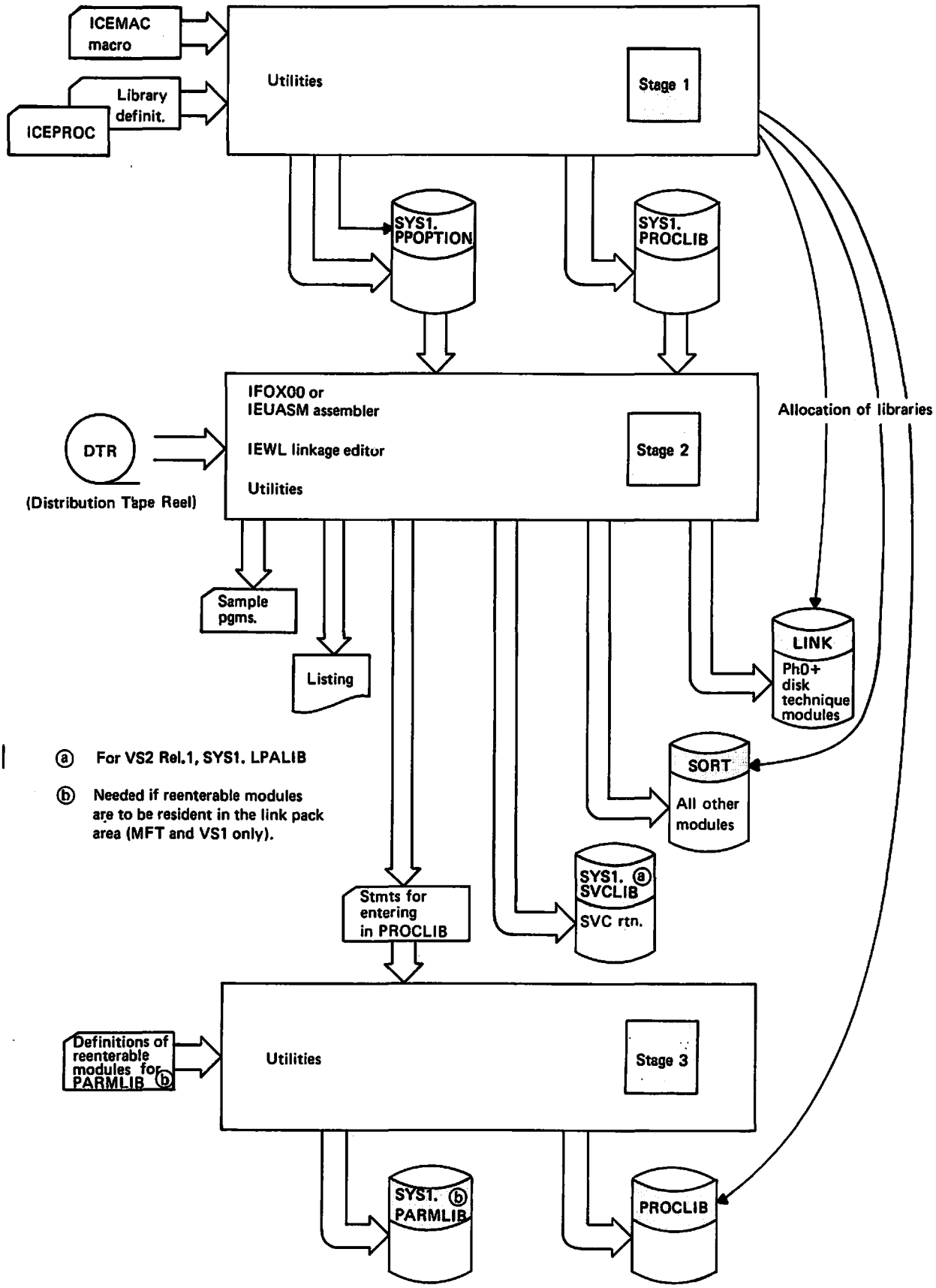


Figure 4. The Installation Procedure

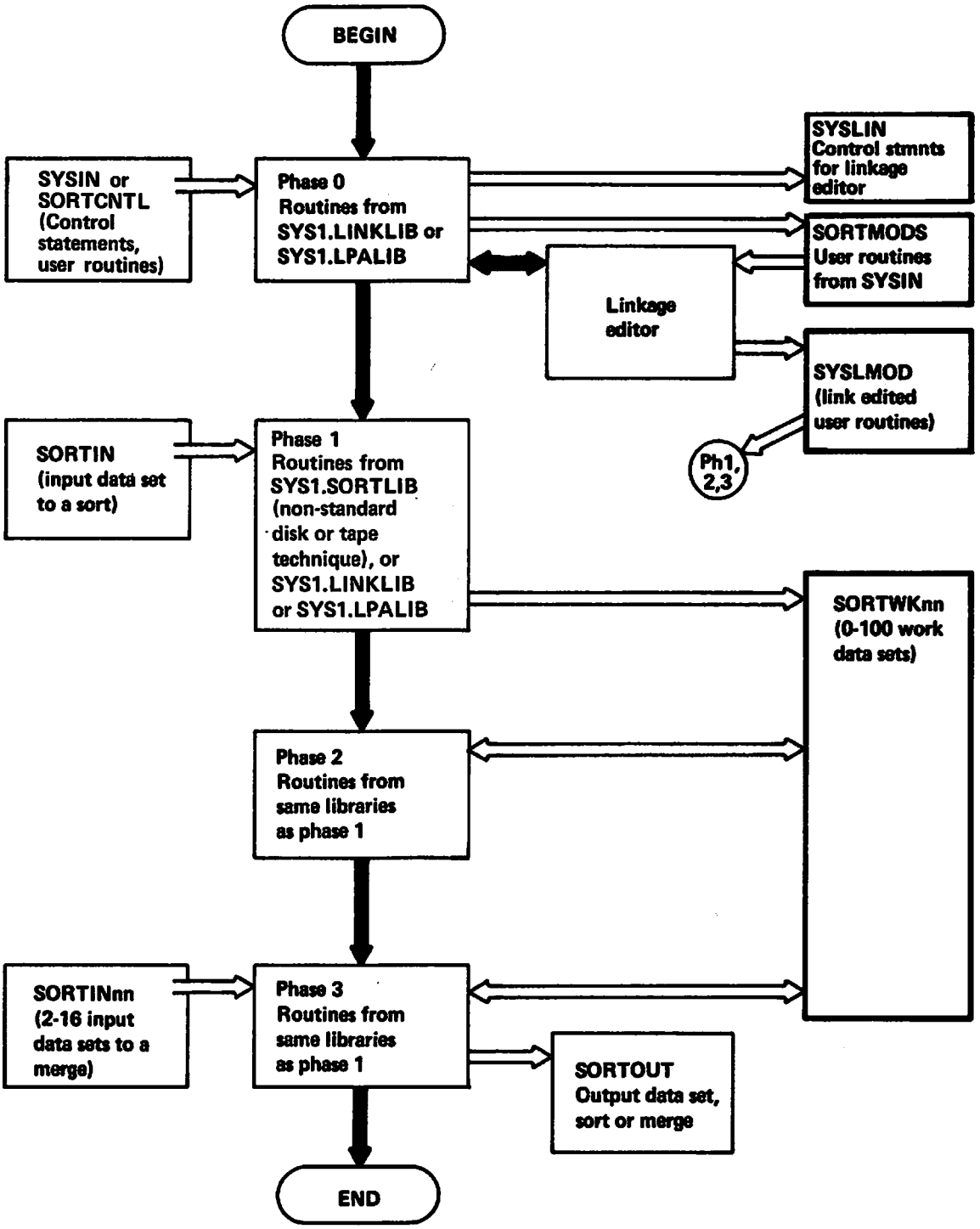


Figure 5. Data Sets Used Only During Execution

Section 2: Method of Operation

This section gives a functional description of the sort/merge program, with cross references to other parts of the manual. It also provides an entry into the microfiche by showing which modules perform which function. It does not, however, describe in detail the various techniques used by the program.

How the Section Is Organized

The section contains information in four forms:

1. A hierarchical overview of the functions of the program, Figure 6, which also serves as a table of contents to the diagrams.
2. The Method of Operation (MO) diagrams, each describing a function of the program in the form input-process-output.
3. Extended descriptions accompanying each lowest-level diagram.
4. Module tables accompanying each lowest-level diagram for conventional techniques, showing which modules perform which function. For peerage and vale modules this information is included in the extended descriptions.

How to Use the MO Diagrams

To find the diagram you need, refer to the overview (Figure 6).

The diagrams are arranged with the processes (functions) listed and numbered in the central block. Start reading in that block, and refer to the left-hand column to find input to each function, and the right-hand column for output. The extended descriptions give more detailed information.

REFERENCES TO DATA AREAS

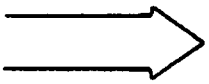
The program makes extensive use of a large data area. For Blockset, the area is called COMMON. See Section 5 for the area layout. For Peerage and Vale appplications the area is called COMMA, and its layout can be found in Section 5. For other applications the information in COMMA is moved in Phase 0 to a 'control phase information area', the CPI, and thence to a 'phase-to-phase information area', the PPI, the DSECT for which resides in module ICERCA.

Virtually all parts of the program use COMMON, COMMA, or the PPI. This is not therefore normally shown in the MO diagrams; exceptions are made when for example updates to COMMON, COMMA, or the PPI, form the major output from a function.

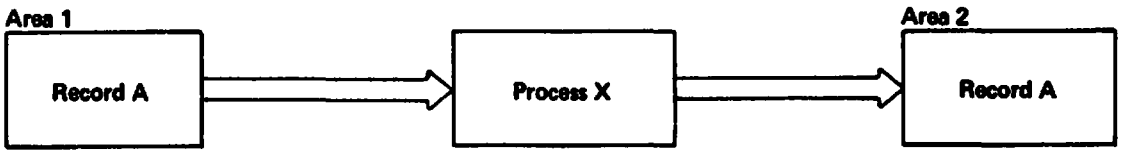
The data areas themselves are described in detail in Section 5 of this publication. To find which specific fields are referenced by a particular CSECT, see the cross-reference list in the relevant microfiche. To find which modules reference a specific field, see the cross-reference tables in Section 6.

SYMBOLS USED

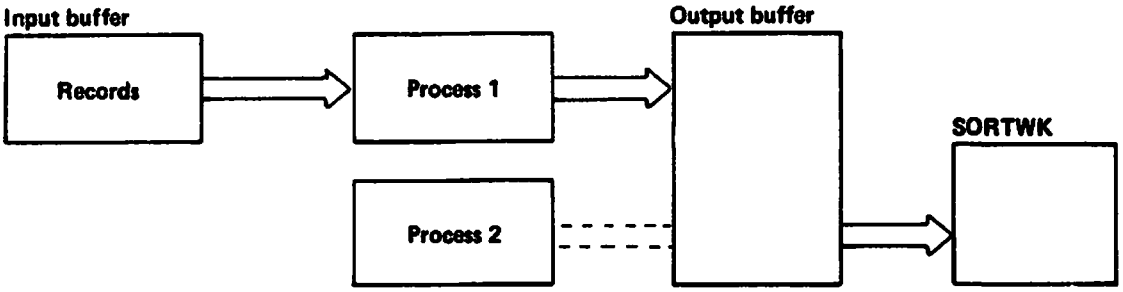
Processing sequence
Entry into a diagram is shown from the previous diagram on the same level; if none exists, the next higher level is shown. Exit from a diagram is shown in the same way.



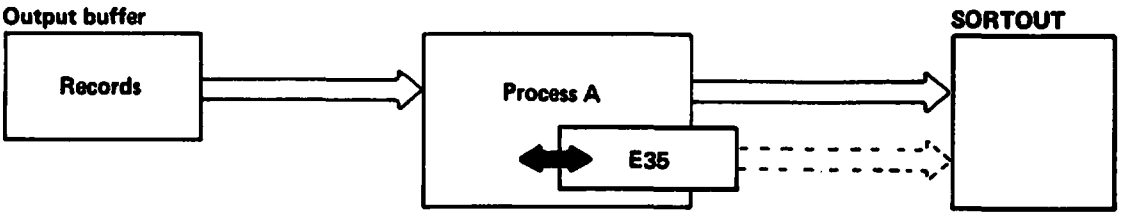
Data movement
For example:



Process X moves record A from area 1 to area 2.



Process 1 moves records from an input buffer to an output buffer. Process 2 moves them from the output buffer to a SORTWK data set.



Process A moves records from an output buffer to the SORTOUT data set. It also passes control to any user routine present at exit E35, which may also move records to SORTOUT.

Pointer - the field from which the arrow originates contains the address of the field or area to which the arrow points. This symbol is also used in the extended descriptions to the diagrams, as follows: '...Exit Name Table (↑PPIAPGC means that a pointer to the Exit Name Table can be found in PPIAPGC.

Data Reference - the process block from which the arrow originates accesses the information in the area to which the arrow points, without modifying it.

Diagram Reference - the functions in the process block are described in greater detail in diagram N.

How to Use the Module Tables

Function

Relates to a function in the process column of the relevant diagram. To find which module carries out the function, follow across to the next column.

Conditions

Work from left to right choosing the conditions which apply to your application.

Mod

Gives the last three or four letters of the name of the module which carries out the function. A directory of modules is given in Section 4.

When a running module carries out a function it alone is referenced, though the corresponding assignment module is of course also necessary to the function. When an assignment module actually carries out the function, it alone is referenced. See Section 4 for more information on assignment and running modules.

ABBREVIATIONS USED IN MODULE TABLES

Fix	fixed-length records
Var	variable-length records
Exit	one or more program exits activated
No E	no program exits activated
256+	records of 256 or more bytes
<256	records of less than 256 bytes
Mult	a multiple compare routine used
Extract	an extract compare routine used
Spanned	output consists of spanned records
Forward	(forwd) read tape forward
Backward	(back) read tape backward

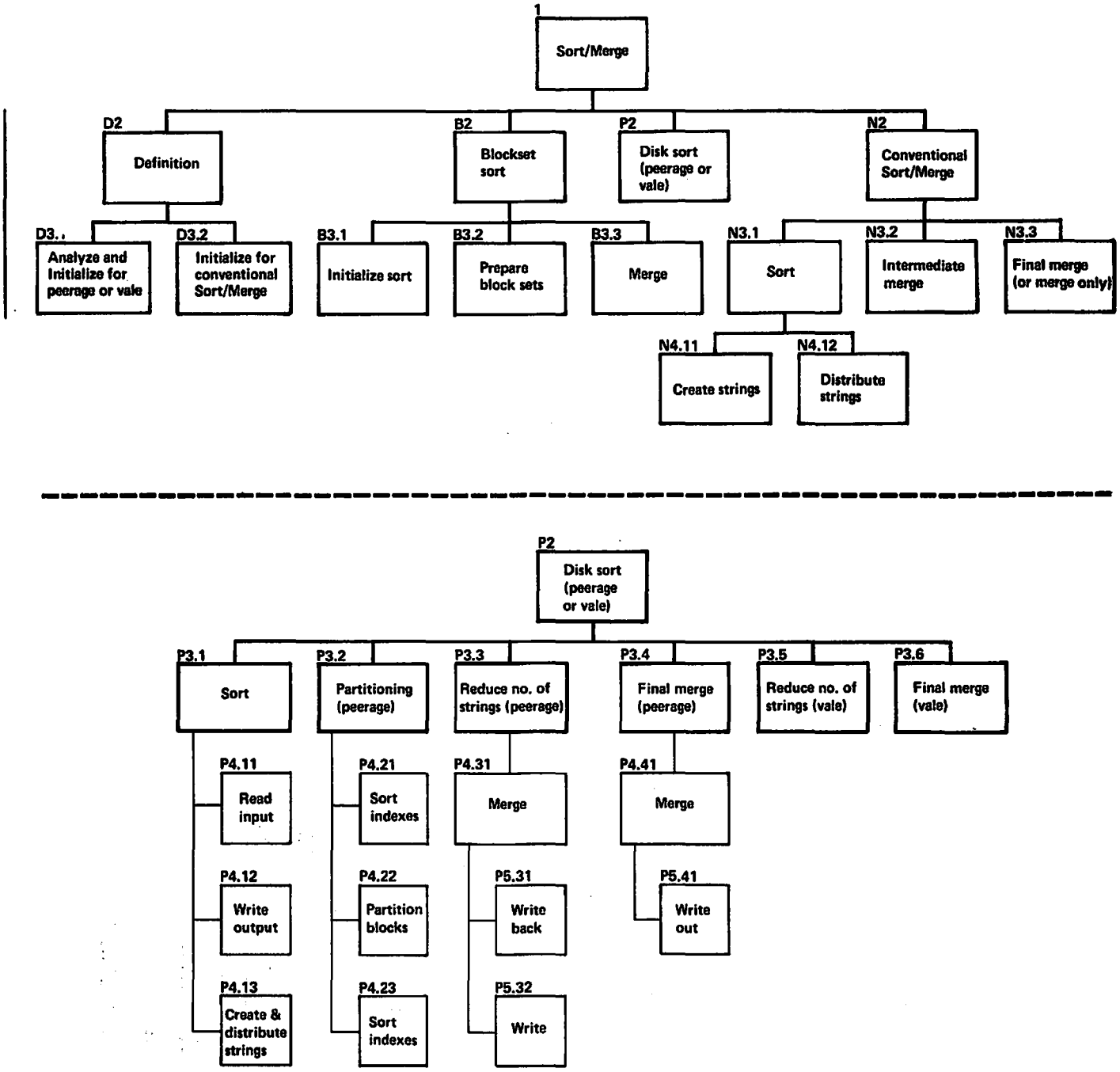
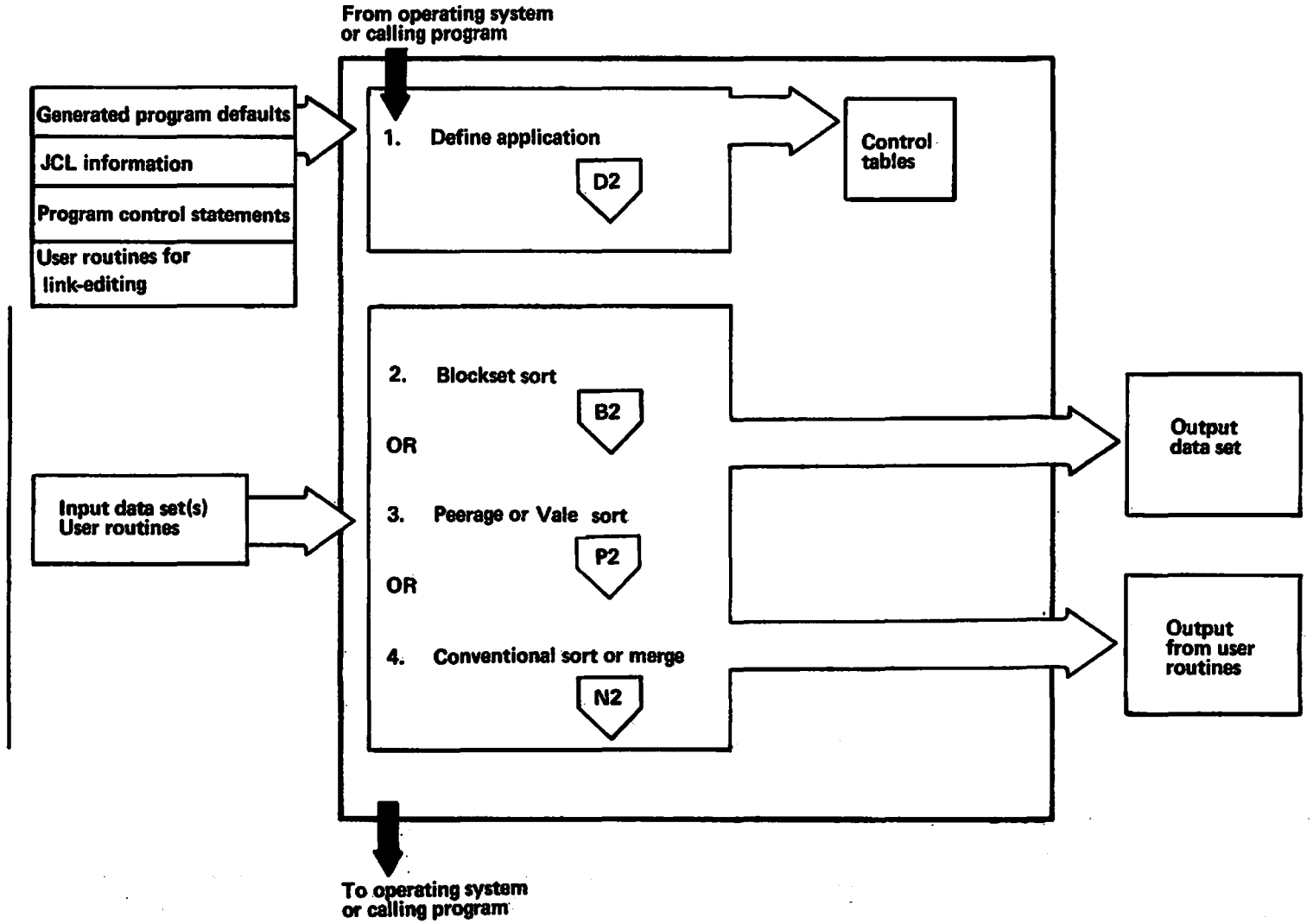


Figure 6. Table of Contents for Method of Operation Diagrams Hierarchical Overview of Program Functions

Sort/Merge



Extended Description for Diagram 1

1. Analyze control information

Determine whether the application is a Blockset, Peerage, or Vale technique sort, or a conventional technique Sort or Merge.

Initialize for the appropriate application.

2. Blockset sort

Sort input into ordered strings, attempting to limit the key range of cylinders.

Create optimal virtual block sets from physical blocks on work file cylinders.

Merge virtual blocks to produce final output.

3. Peerage or Vale sort

Sort input into ordered strings, using the replacement selection technique.

For Peerage, partition the work files. The partitioning technique is shown in MO diagram P3.2.

Reduce the number of strings.

Merge the strings to produce final output.

4. Conventional Sort

Sort input into ordered strings, using the replacement selection technique.

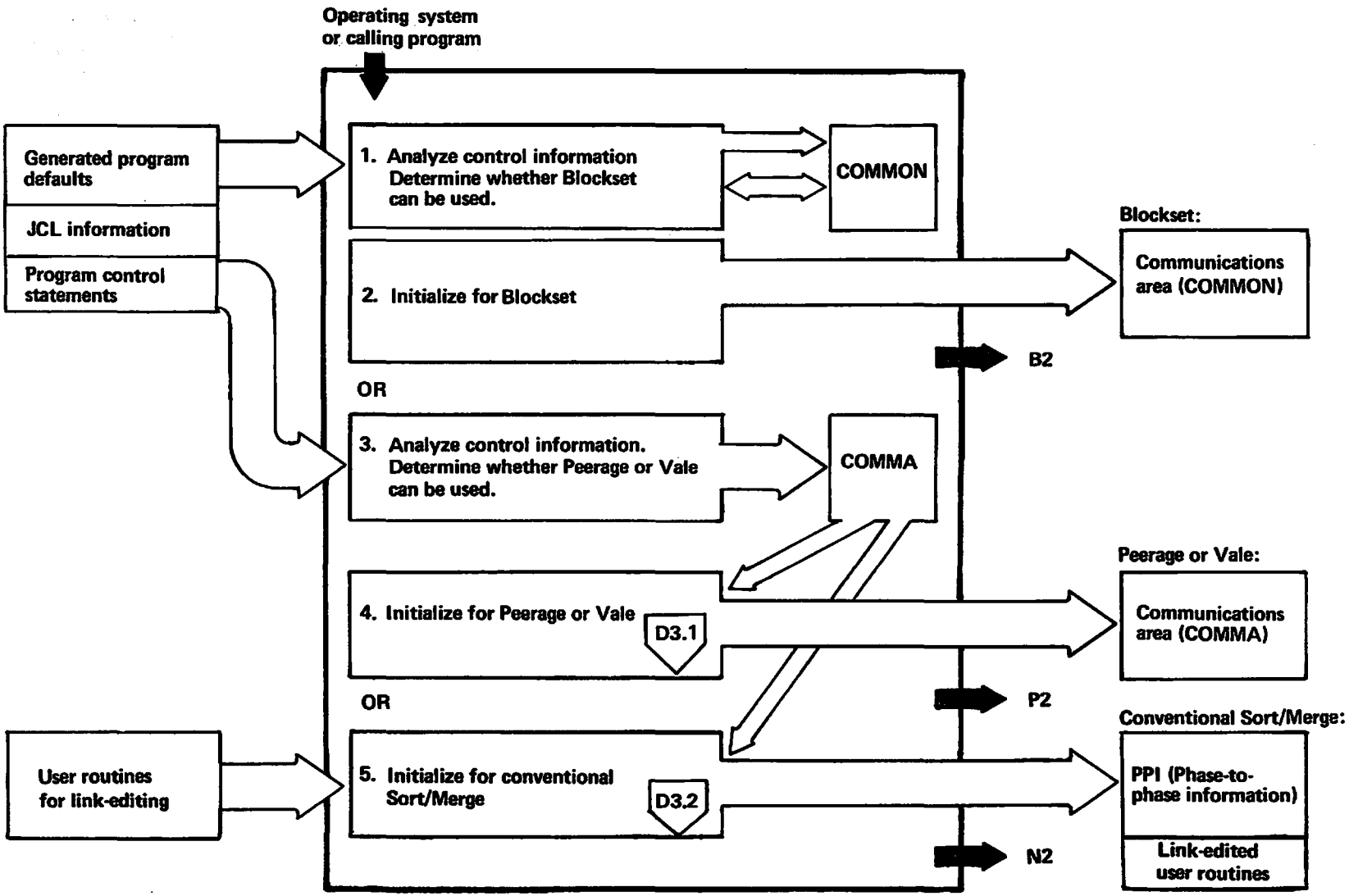
Reduce the number of strings.

Merge the strings to produce final output.

Merge

Merge the input files to produce final output.

D2 Definition



Extended Description for Diagram D2

1. Process program options (default and specified), JCL information, and program control statements.

   Optimize control field specifications.

   Check Blockset technique criteria (listed in Section 1 under 'Program Organization').

2. Initialize for Blockset:

   Generate code to handle control fields.

   Make the program reenterable.

   Set up control block areas.

   Optimize.

3. Process program options (default and specified), JCL information, and program control statements.

   Optimize control field specifications.

   Check Peerage technique criteria (listed in Section 1 under 'Program Organization') to determine whether this is a Peerage technique sort; if the criteria are not met, check Vale criteria (also listed in Section 1).

4. Initialize for Peerage or Vale Sort:

   Generate code to handle control fields.

   Make the program reenterable.

   Set up control block areas.

   Optimize.

5. Initialize for conventional Sort or Merge:

   Build CPI.

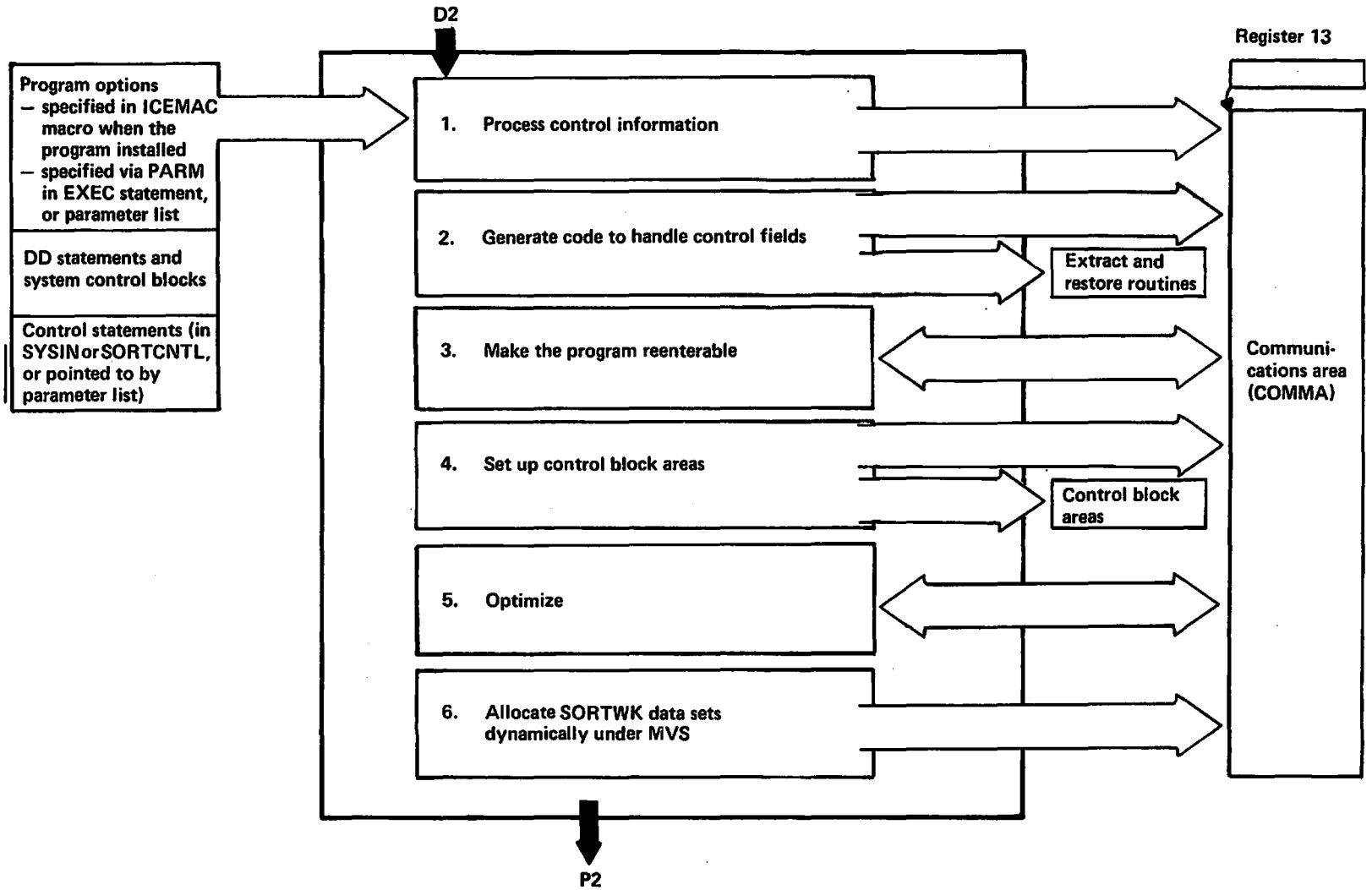
   Prepare user routines.

   Set up program control areas.

   Optimize.

D3.1

Analyze and Initialize for Peerage or Vale



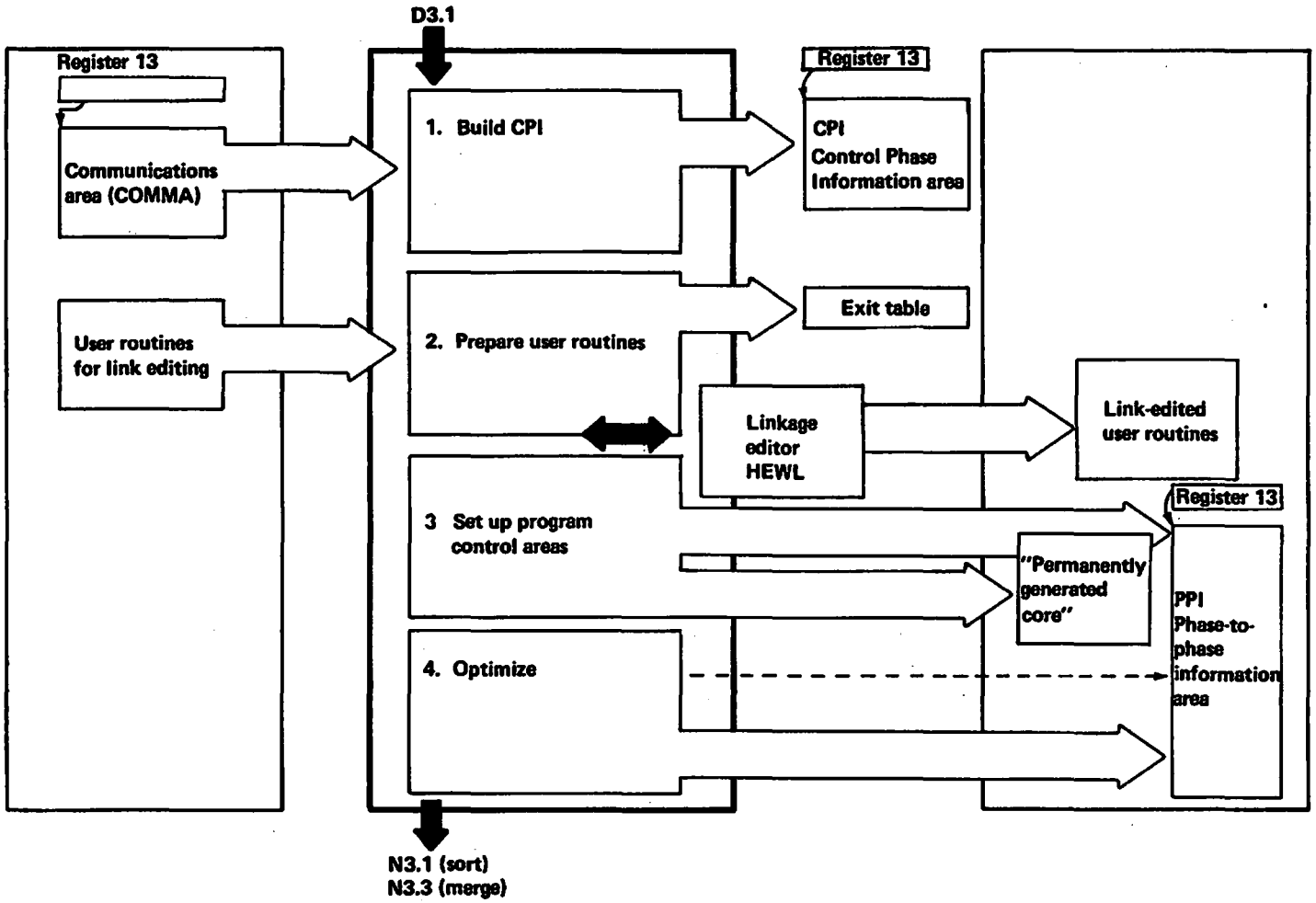
Extended Description for Diagram D3.1

	Module
1. a. Check generated defaults (stored in Module ICEAM1, described in Section 5).	DEF
Scan PARM field parameters from EXEC statement, or (for an invoked sort) the parameter list, pointed to by register 1.	DEF
Check that no invalid options selected.	
Make appropriate entries in the PPI (see the 'Index to PPI Fields' in Section 5).	
b. Check that necessary DD statements are present, and are not duplicated.	DEF
Open SORTIN, SYSOUT, and SYSIN or SORTCNTL data sets, and check for valid DCB parameters.	DED
c. Read program control statements, from SYSIN, from SORTCNTL, or from addresses given in the parameter list. Check the statements for correctness and completeness.	DEC
2. Generate in main storage a routine for transposing records before sorting them, and for restoring them to their original format after sorting them.	DEG

	Module
3. Move code which is not reenterable to a reserved area: - tree handling routines - special code	DEG
4. Create a control block area for each SORTWK data set. The area holds IOB, DCB, ECB, link fields, and other information.	DEG
5. Determine: - no. and type of input buffers - internal blocksize - index size - maximum M for phases 2 and 3	DEV
Optimize the use of the intermediate storage data sets.	
Optimize the mix of SORTIN/SORTOUT buffers and SORTWK buffers, given the way in which the files are allocated, the data transfer rate of the devices, etc.	DEV
6. Allocate the number of specified (or defaulted) work areas with the space calculated or using the system's default value. Reallocate VIO data sets with a DSNAME specified, if required.	DEG

D3.2

Initialization for Conventional Techniques



Extended Description for Diagram D3.2

1. Move information from the communications area to the CPI. Release the communications area. If SIZE(MAX) is in operation, calculate how much virtual storage is available for sorting purposes, and note the amount in the CPI. The CPI is described in Section 5.

2. If user routines are present:

   a. Copy any routines in SYSIN to SORTMODS.

   b. If necessary, prepare linkage editor control statements, and put them in SYSLIN.

   c. If necessary, create a parameter list for the linkage editor.

   d. Build an exit name table for exits that do not require further linkage editing.

3. a. Transfer the information in the CPI to the PPI. The PPI is described in Section 5.

   b. Get the storage necessary for the 'permanently generated core' area, to contain the exit name table, the format track table, and the control field handling routine.

   c. Generate the appropriate control field handling routine.

4. For a Sort

   a. Choose the sequence distribution technique to be used.

   b. Calculate bin size.

   c. Calculate variables:

      - B (sort blocking) and G (RSA capacity)
      - maximum M (defined in Section 3 under 'Phase 2')
      - the number of input and output buffers to be used.

   For a Merge

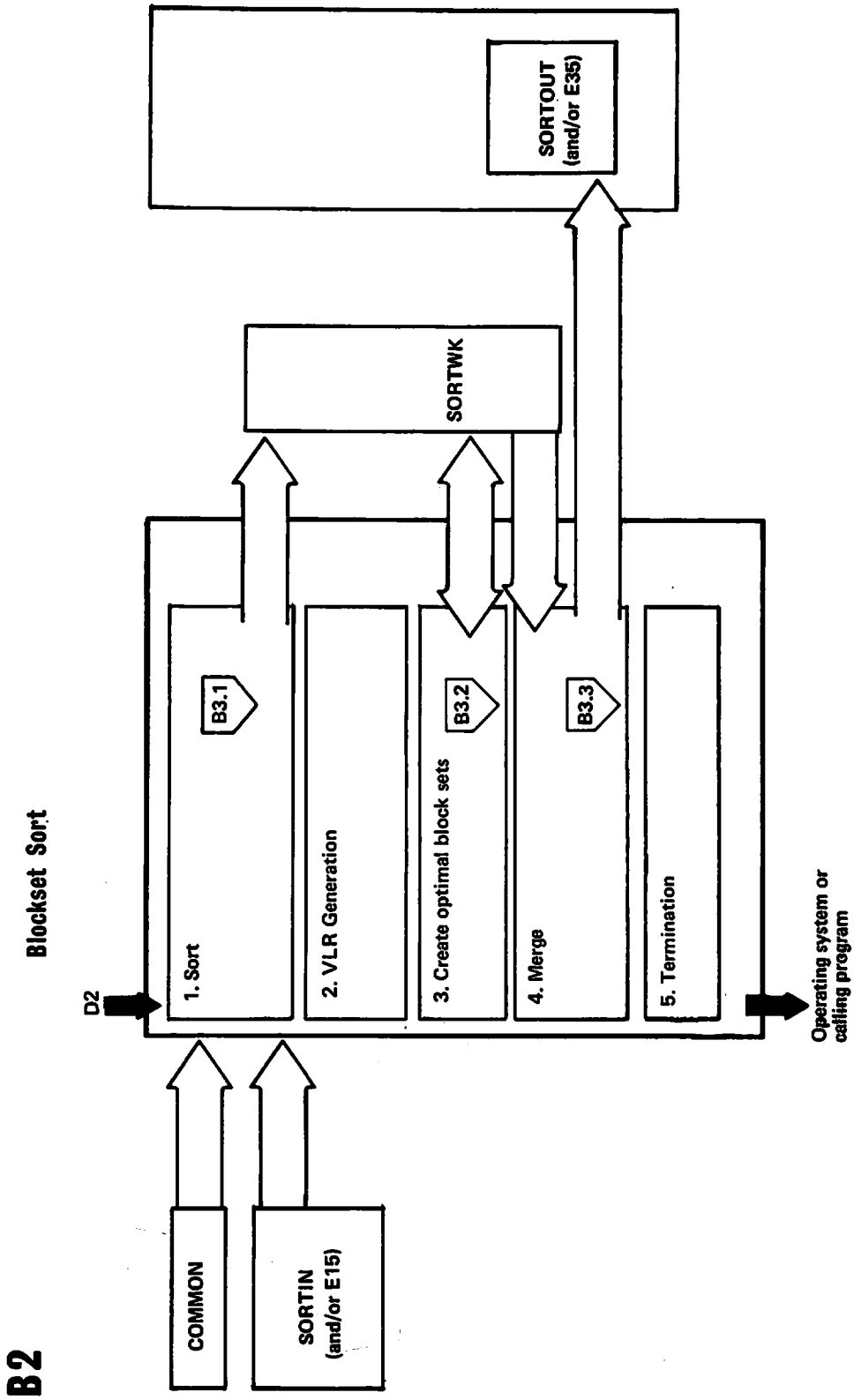
   d. Determine the number of input and output buffers to be used.

Module Table for Diagram D3.2

Function	Conditions		Mod
1 Create CPI	Always		RCM
2a Copy rtns	Always		RCH
b Link ed. stmts	Always		RCP
c Parameter list	Always		RCM
d Name table	Always		RCH
3a Create PPI	Always		RC1
b Get storage	Always		RC1
c Gen multiple rtn	Always		AOL
Gen extract (1)	Always		AOM
Gen extract (2)	E61 activated*		AO5
	Otherwise		AO4
4a Choose technique	Tape		RCR RCS BGA
	Disk		RCN
b Calc bin size	(As for 4a)		
c Calc variables	Tape (As for 4a)		
	Disk	BALN	RCK
		CRCX	BGB
d Calc no. of merge buffers	Always		RCL

*And/or control fields of type AC,AQ,CLO,CSL,CST.

This page intentionally left blank


B2
Blockset Sort
COMMON
SORTIN (and/or E15)
SORTWK
SORTOUT (and/or E35)
D2
1. Sort
2. VLR Generation
3. Create optimal block sets
4. Merge
5. Termination
B3.1
B3.2
B3.3
Operating system or calling program

Extended Description for Diagram B2

1. Read input via input buffers into the RSA (Record Storage Area).

   FLR: If end of SORTIN is reached when few or no blocks have been written to SORTWK, bypass the key handling phase. Otherwise, write records to SORTWK to make room for new records to be read into the RSA. Index the blocks written to SORTWK.

   VLR: Write records to SORTWK to make room for new records to be read into the RSA. Collect record length data to use in computing bin size. Index the blocks written to SORTWK. At end of SORTIN, write remaining records to SORTWK and all index records.

2. VLR Generation Phase: Compute bin size and generate code used by output phase to restore records.

3. Sort the indexes created by the SORT phase, and create optimal virtual block sets (physical blocks on same cylinder) for the merge phase.

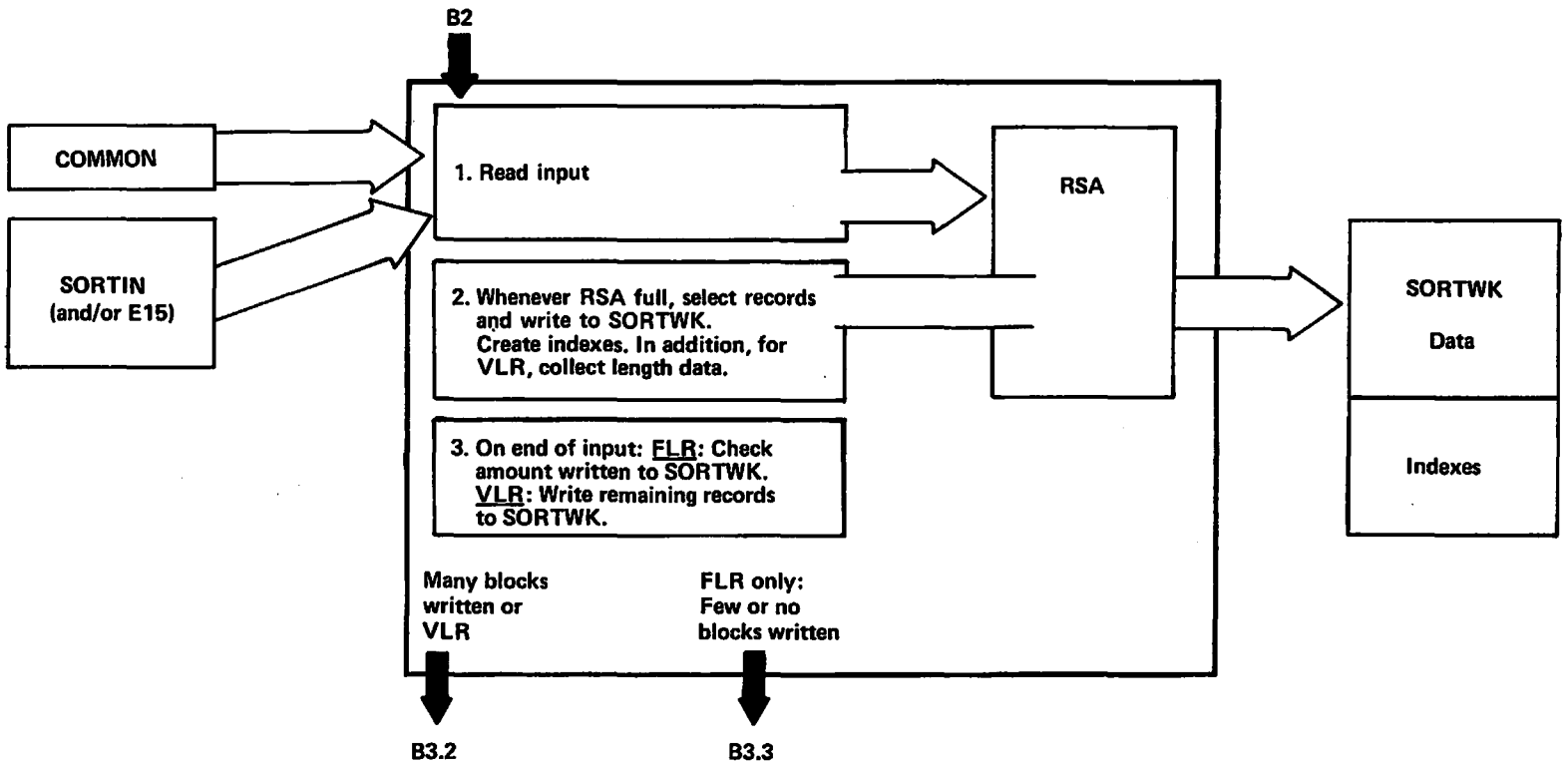
4. Merge virtual block sets to create the output data.

5. Terminate:

   Print information messages.
   Close files used.
   Free storage used.

B3.1

Initialize Sort

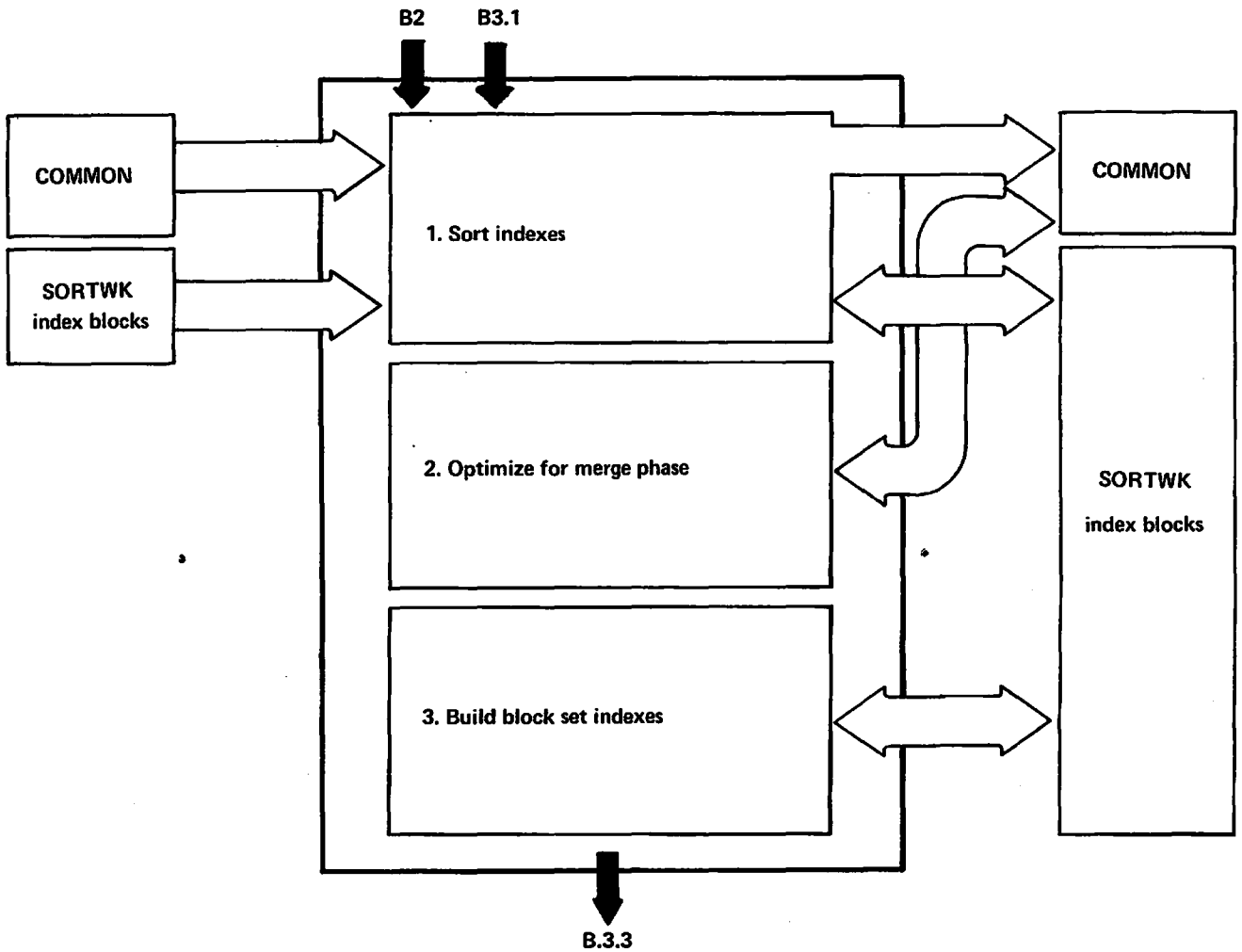


Extended Description for Diagram B3.1

	Module
1. Get input records and store, partially sorted, in the Record Storage Area (RSA).	IPUM IPVM
2. a. From partially sorted RSA records select an output block continuing output string, or if no records for current output string begin a new one.	IPUM IPVM
b. Select a cylinder on SORTWK data sets such that the keys in the block to be written are close to keys already written on cylinder.	IPUB IPVB
c. Write selected records to SORTWK, freeing RSA space for more input records.	IPUB IPVB
d. Make block indexes	IPUB
3. FLR: If no SORTWK space or one string of data written to SORTWK, skip the key handling phase and go to Merge phase.	IPUT
VLR: At input end of file, write remaining records and indexes to SORTWK. Call ICEGENV to initialize code for the OUTPUT phase.	IPVT

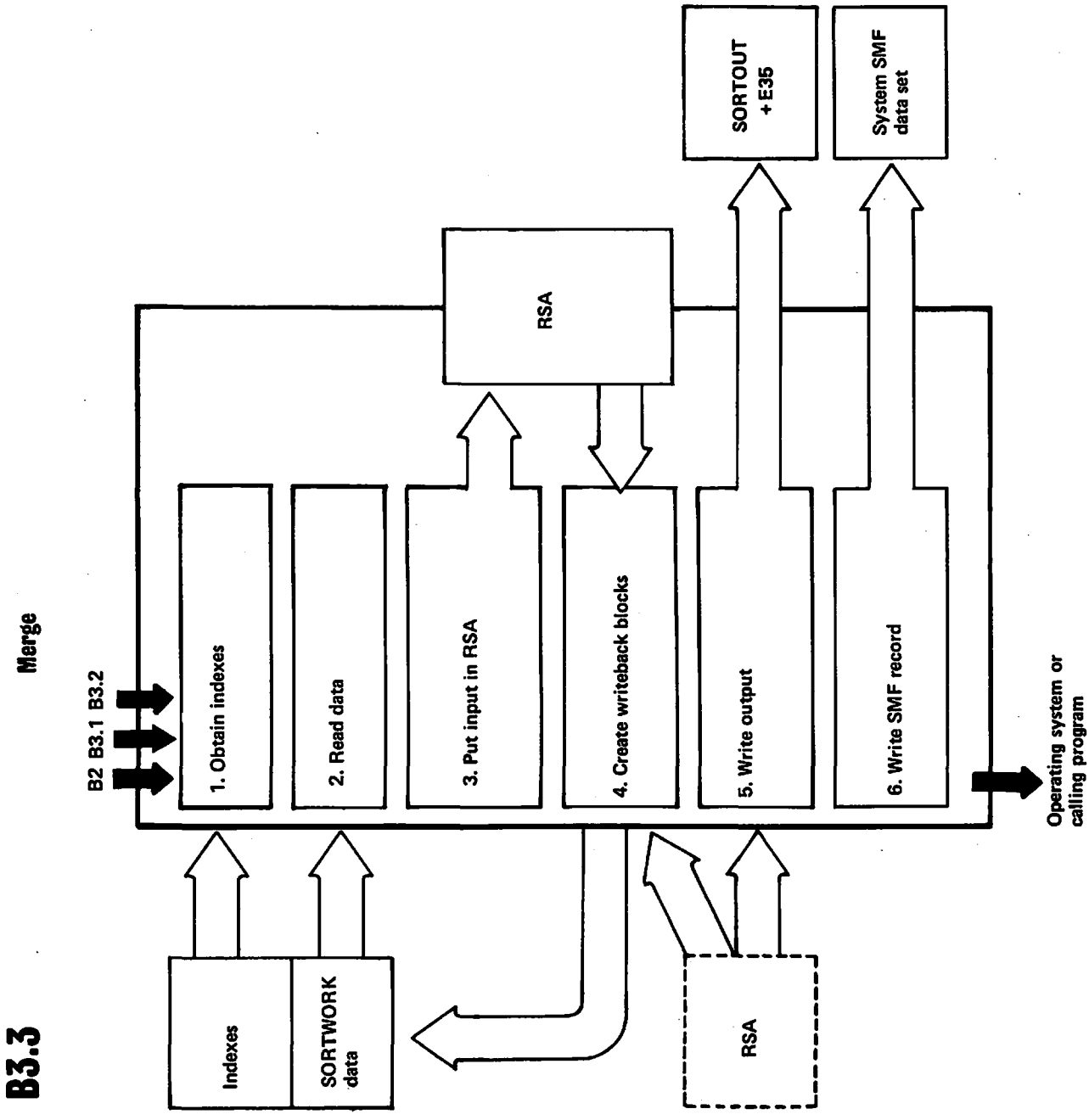
B3.2

Key Handling Phase (Prepare Block Sets)



Extended Description for Diagram B3.2

	Module
1. Sort the indexes by recursive application of the writeback merge algorithm of the merge phase.	KPUS KPVS
2. Determine virtual block set size, optimize for next phase:	KPUT KPVT
FLR: If the number of strings created by sort is small enough to allow merging without write-back, largest possible block set size is selected.	KPUV
Otherwise a smaller block set size is selected.	KPUV
3. Create virtual block indexes: Process indexes of physical data blocks in key order, and group those on same cylinder with key values sufficiently close together to form virtual block sets.	KPUV KPVV

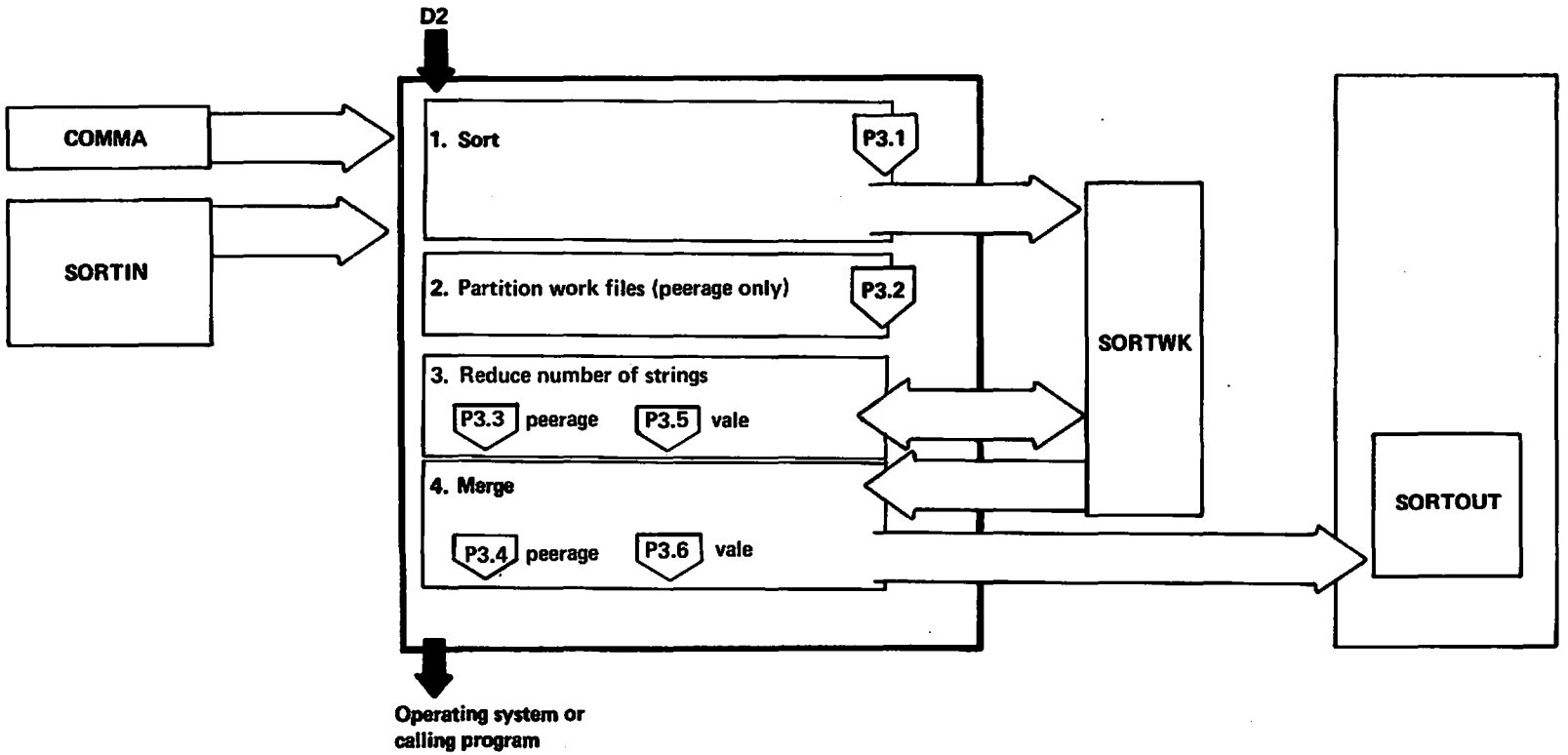

B3.3
Merge
B2 B3.1 B3.2
1. Obtain indexes
2. Read data
3. Put input in RSA
4. Create writeback blocks
5. Write output
6. Write SMF record
RSA
SORTOUT + E35
System SMF data set
Operating system or calling program
Indexes
SORTWORK data
RSA

Extended Description for Diagram B3.3

	Module
1. Obtain (next) input virtual block index. If necessary read indexes from SORTWK, and/or build new virtual block indexes from writeback physical block indexes. If enough space in RSA for a buffer full of input, continue. Otherwise skip point 3.	OPUT OPVT
2. Read data blocks. Obtain disk addresses of virtual block set and read those in optimal order into input buffers.	OPUT OPVT
3. Unload input buffers into RSA. Return to point 1. If enough space can be created by moving RSA records to output, skip to point 5.	OPUA OPVA
4. Select records from input buffers and RSA to form part of a writeback string, and write the string to SORTWK in the same way as during phase 1 (sort): Select suitable writeback cylinder, and create indexes for writeback blocks. If necessary spill indexes to SORTWK index blocks.	OPUT OPVT
5. Transfer records to output. If RSA and input buffers empty, finish processing. Otherwise return to point 3.	OPUA OPVA
6. Build and write SMF record if requested (VS1, MVS only).	EXIO OXOV

P2

Peerage and Vale Sort



Extended Description for Diagram P2

1. Read input via input buffers into the RSA (Record Storage Area).

   If end of SORTIN is reached before the RSA is full, SORTOUT can be written immediately. The tree is used to select records for output.

   Otherwise records are written out to SORTWK to make room for new records to be read into the RSA. For Peerage the blocks written to SORTWK are indexed.

Peerage

2. Sort the indexes, as described in MO diagram P4.21.

   If the number of logical strings is <maximum M for phase 3, control is passed to the final merge (shown in MO diagram P3). M (merge order) is explained in Section 3 under 'Phase 1'.

   If there are more strings, an intermediate merge pass is needed.

   Partition the blocks on SORTWK, and arrange the indexes to reflect the partitioning, as described in MO diagrams P4.22 and P4.23.

3. Merge all partitions except partition 0, creating new strings on SORTWK.

   Return control to point 2.

4. All strings are now in partition 0.

   Merge partition 0 to produce the output data set.

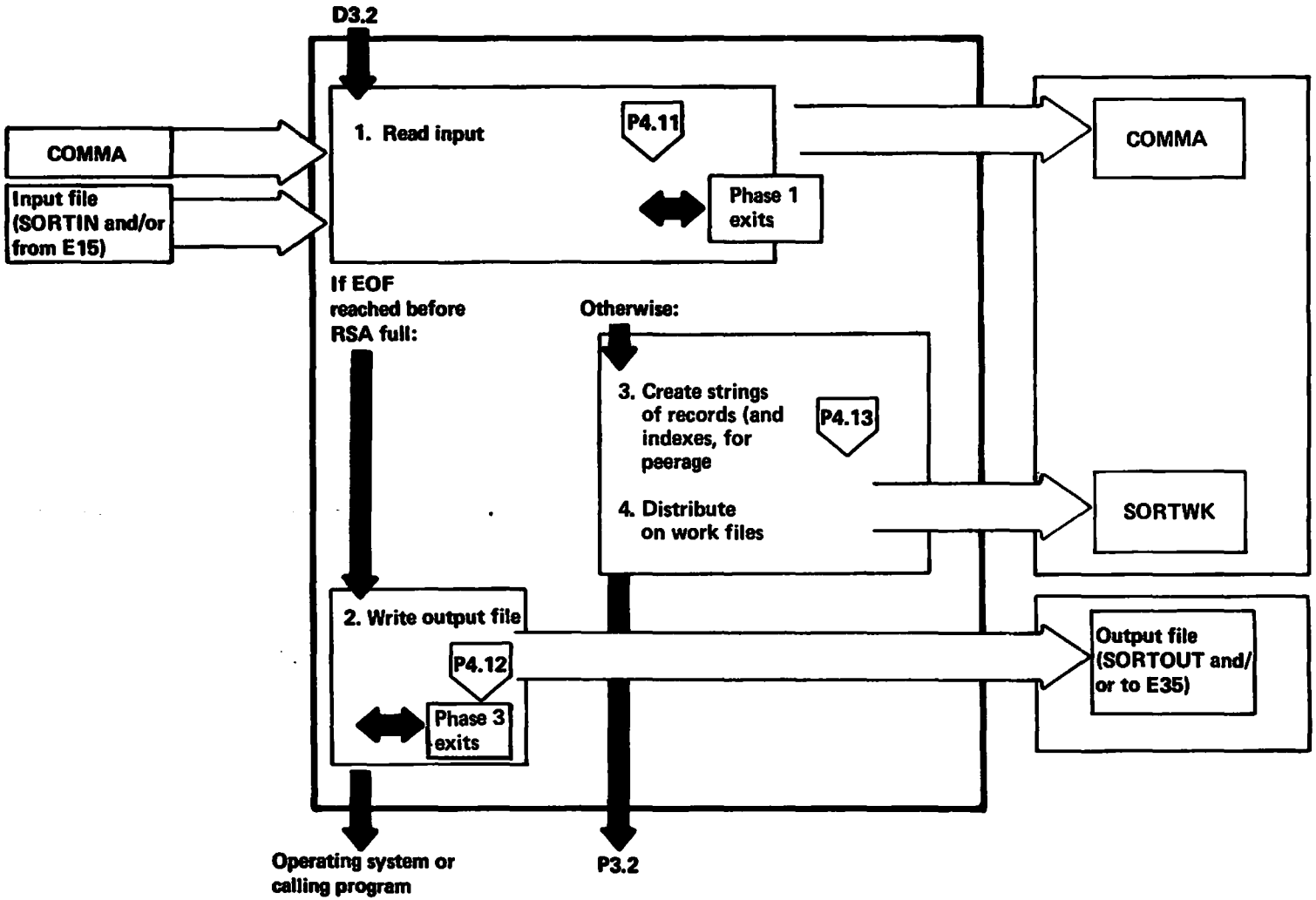
Vale

2. (Not used.)

3. Merge as many strings as possible (M for phase 2). Repeat until the number of strings is ≤ phase 3 merge order.

4. Merge the remaining strings to produce the output data set.

P3.1 Sort



Extended Description for Diagram P3.1

1. Initialize buffers, RSA and tree, and SORTIN.

   Move records into the RSA and update the tree for each record.

   Continue until the RSA is full (or end of SORTIN is reached).

2. If end of SORTIN is reached before the RSA is filled, write the output file.

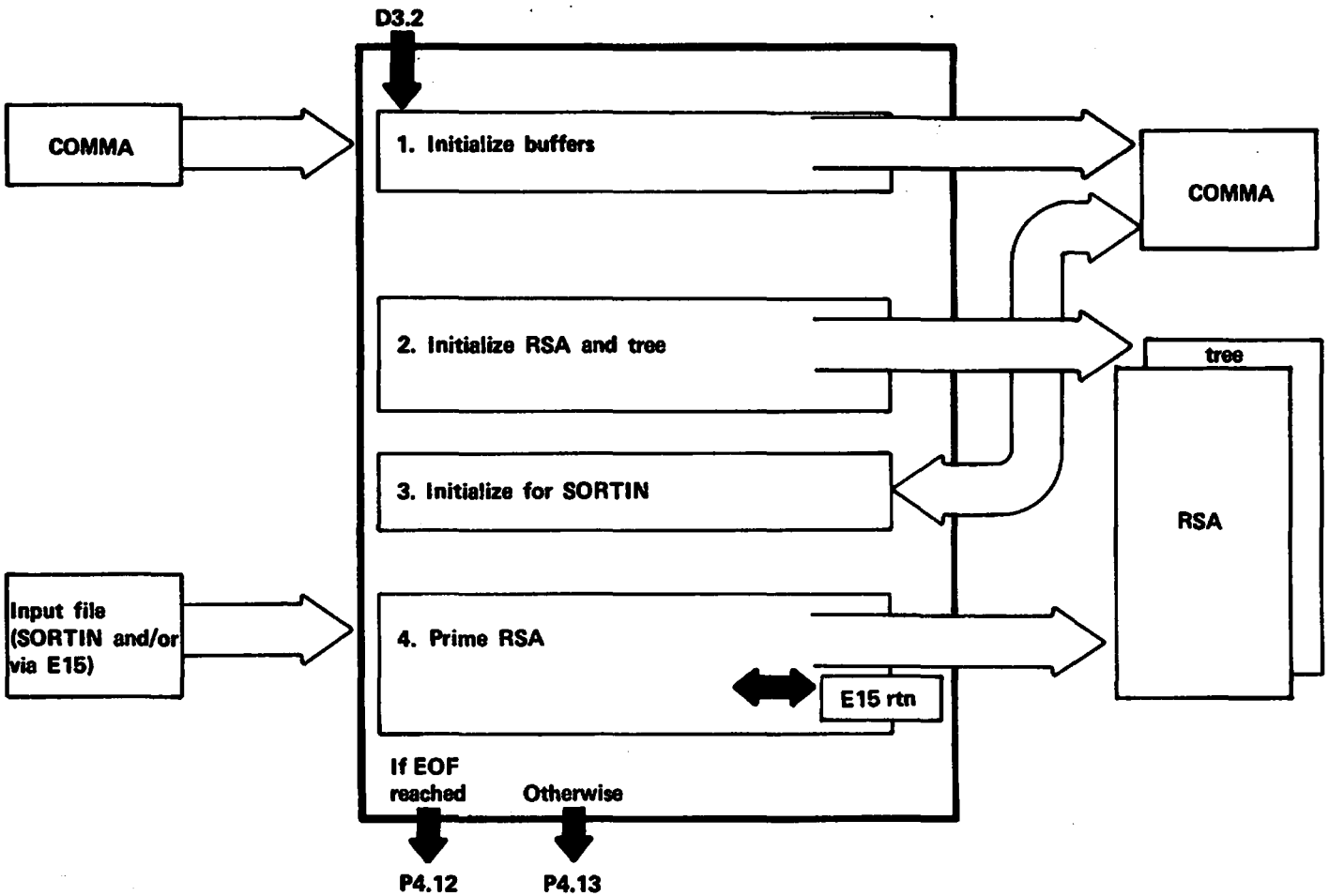
3. Otherwise, intialize SORTWK data sets.

   Create strings of records (and, for Peerage, indexes) for each output block.

4. Write the strings (and indexes) to SORTWK.

P4.11

Read Input



Extended Description for Diagram P4.11

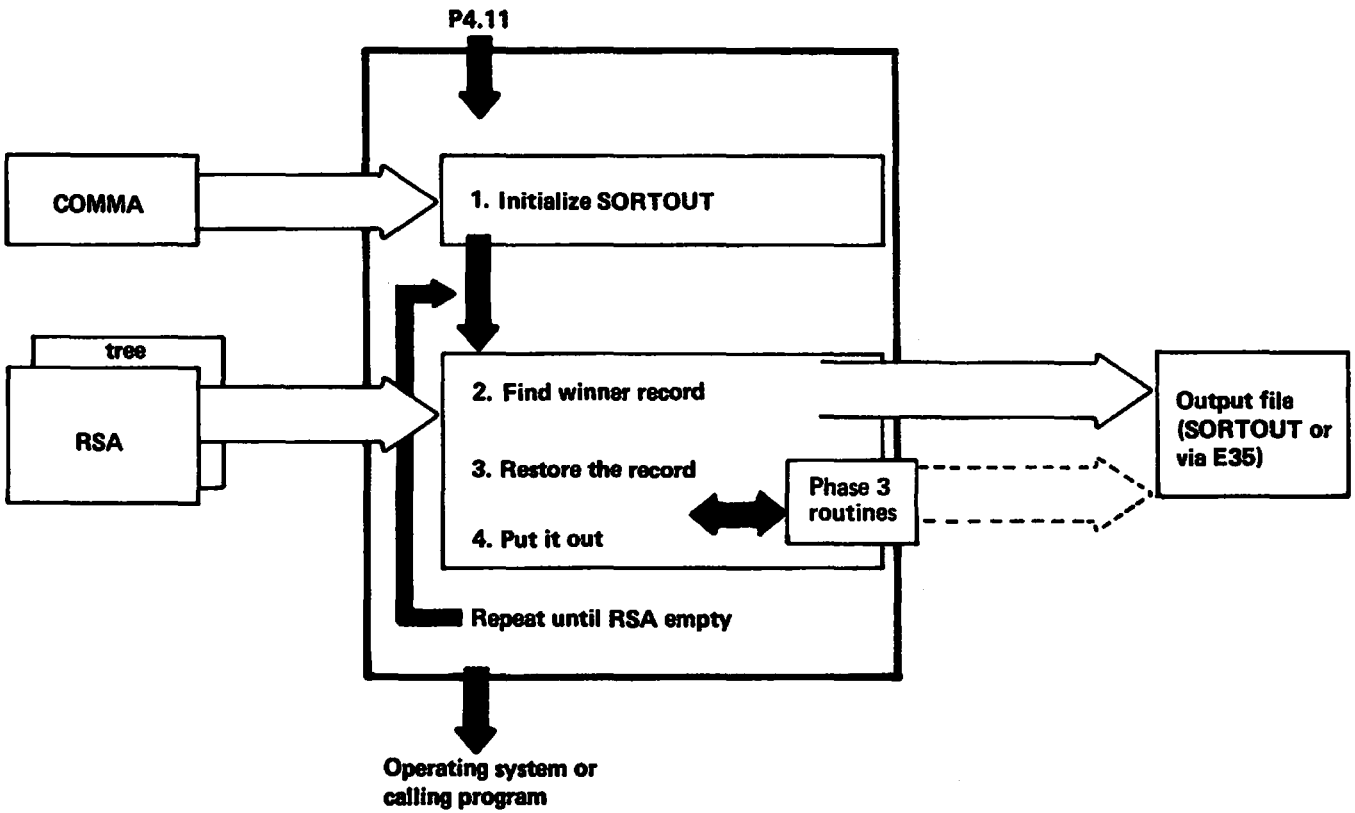
Module	
CRE/VRE	1. Allocate and initialize buffers for SORTWK data sets. Chain the buffer areas together. Allocate and initialize buffers used for SORTIN. (The same buffer area is used for SORTOUT if written in this phase.)
CRE/VRE	2. Allocate all remaining main storage to the RSA and tree. Chain bins in the RSA.
CRE/VRE	3. Initialize addresses for GET and EOF routines. Load phase 1 routines if specified.
CRE/VRE CRE/VRE* DEG*	4. Get a record (from the input buffer or from E15). Transpose the record. Put the record in the next vacant bi in the RSA. Put the address of the record in the correct place in the tree. Repeat until the RSA is full.
DEG*	

If EOF is reached before the RSA has been filled, the records do not need to be written to a SORTWK file. Instead, the SORTOUT file is written, using the tree to locate successive winner records in the RSA.

*The code is generated outside CRE/VRE's area; the source code is held in ICEDEG.

P4.12

Write Output

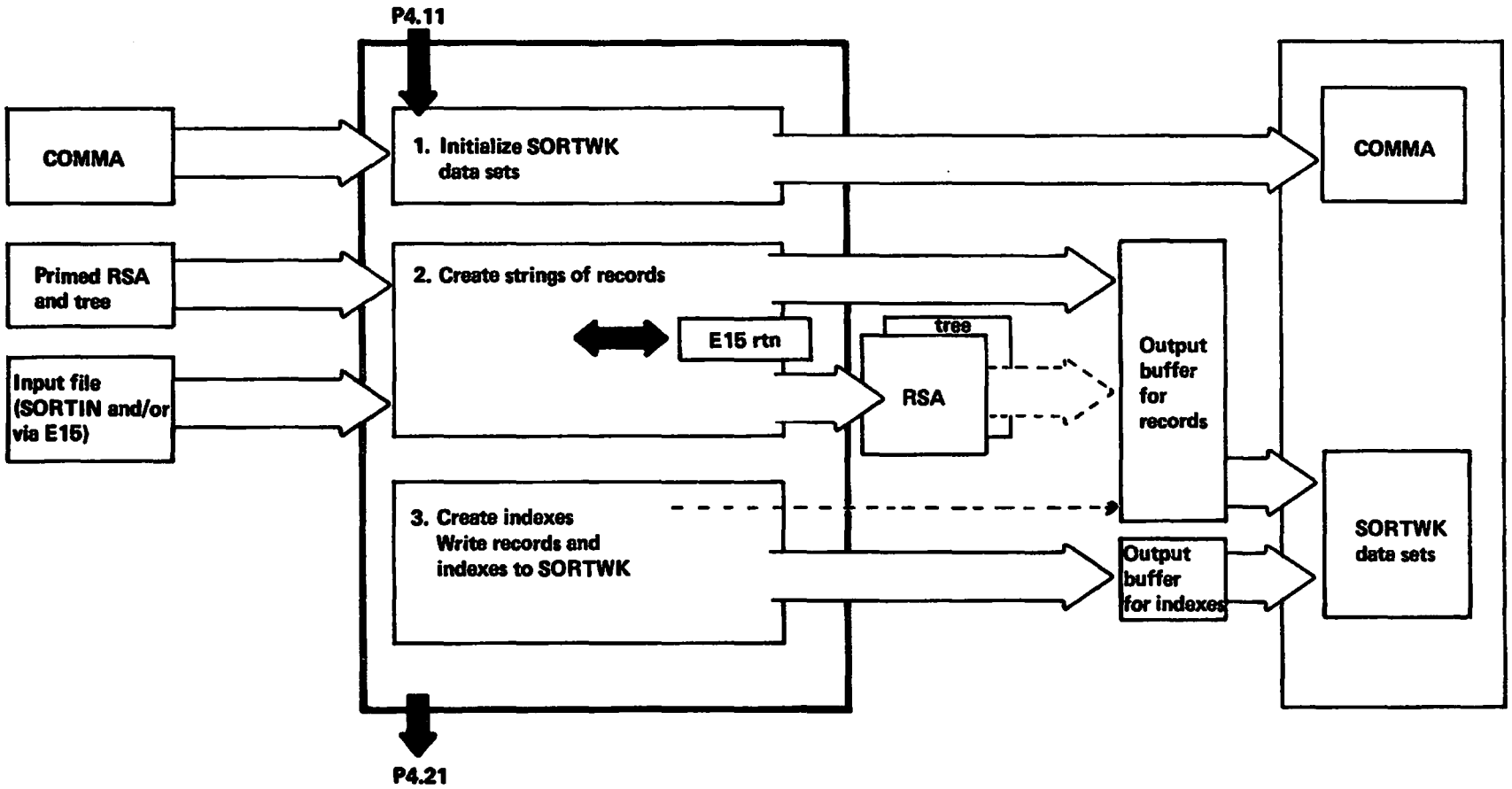


Extended Description for Diagram P4.12

	Module
1. a. Allocate and initialize buffers for SORTOUT. Initialize address for the appropriate put routine. Load E35 routine (if any).	CRO/VRO
b. Open SORTOUT if it is not a VSAM data set. Close the SORTIN data set.	
2. Use the tree to locate the winner record in the RSA.	
3. Restore the record to its original format. Move the record to the output buffer (or pass it to the E35 routine).	*CRO/VRO
4. a. If VSAM, put the record to SORTOUT.	CRO/VRO
b. If non-VSAM, repeat 2 and 3a until the output buffer is full. Write the buffer contents.	
Repeat points 2-4 until the RSA is empty.	CRO/VRO/XCP

*The code which carries out these functions is generated outside ICECRO/VRO's area, and resides in ICEDEG.

P4.13 Create and Distribute Strings



Extended Description for Diagram P4.13

1. Set EOF and PUT routine addresses.

2. Move the winner record from the RSA to the output buffer.
   Read the next record from the input buffer.
   Transpose the record.
   Move the record to the bin in the RSA from which the previous winner was moved.
   Update the tree.

3. a. Not used for Vale. Create a start index for the logical block, using the control word from the record in the output buffer. Put the index in the output buffer for indexes.

   b. Repeat point 2 until the output buffer is full.

   c. Write the block to SORTWK.

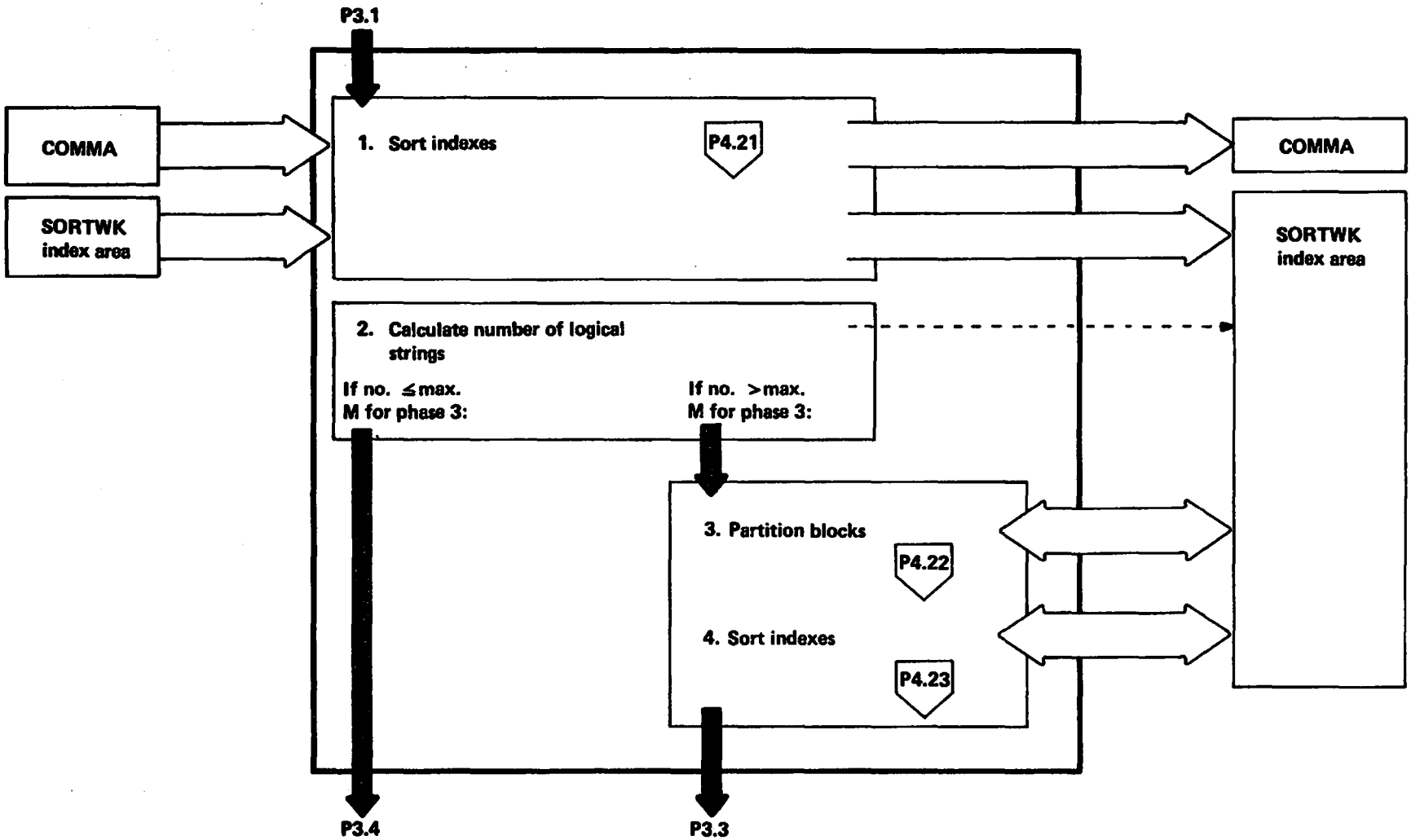
   d. Not used for Vale. Create an end index for the logical block, using the control word from the last record moved to the output buffer. Put the index in the output buffer for indexes.

   e. Not used for Vale. Repeat points 2 and 3a-e until the end of a string is reached. At end of string, make up a complete logical block, i.e. fill the last track with dummy physical blocks, so that the next string will begin a new track.

   Repeat points 2 and 3 until EOF is reached.

Module Peerage	Vale
CRE	VRE/VRN
CRE	VRE/VRN
CRE	VRE/VRN
CRE	VRE/VRN
DEG	DEG
CRE	---
CRE	VRE/VRN
CRE	
CRE	

P3.2 Partitioning



Extended Description for Diagram P3.2

Peerage only (not used for Vale).

1. Read the indexes. Sort them into strings. Index the strings, in the same way as record strings are indexed. Write the indexes to SORTWK. If necessary, merge the higher levels of indexes, until the original indexes are all in one string on SORTWK.

2. In calculating the number of logical strings, physical strings are ignored: only logical blocks are considered. The basis of the algorithm used is that two physical blocks whose contents overlap must necessarily belong to different logical strings. Two blocks whose contents do not overlap, however, can be included in the same string. The indexes can be used to determine whether or not blocks overlap, since they contain the control words of the highest and lowest records contained in the block.

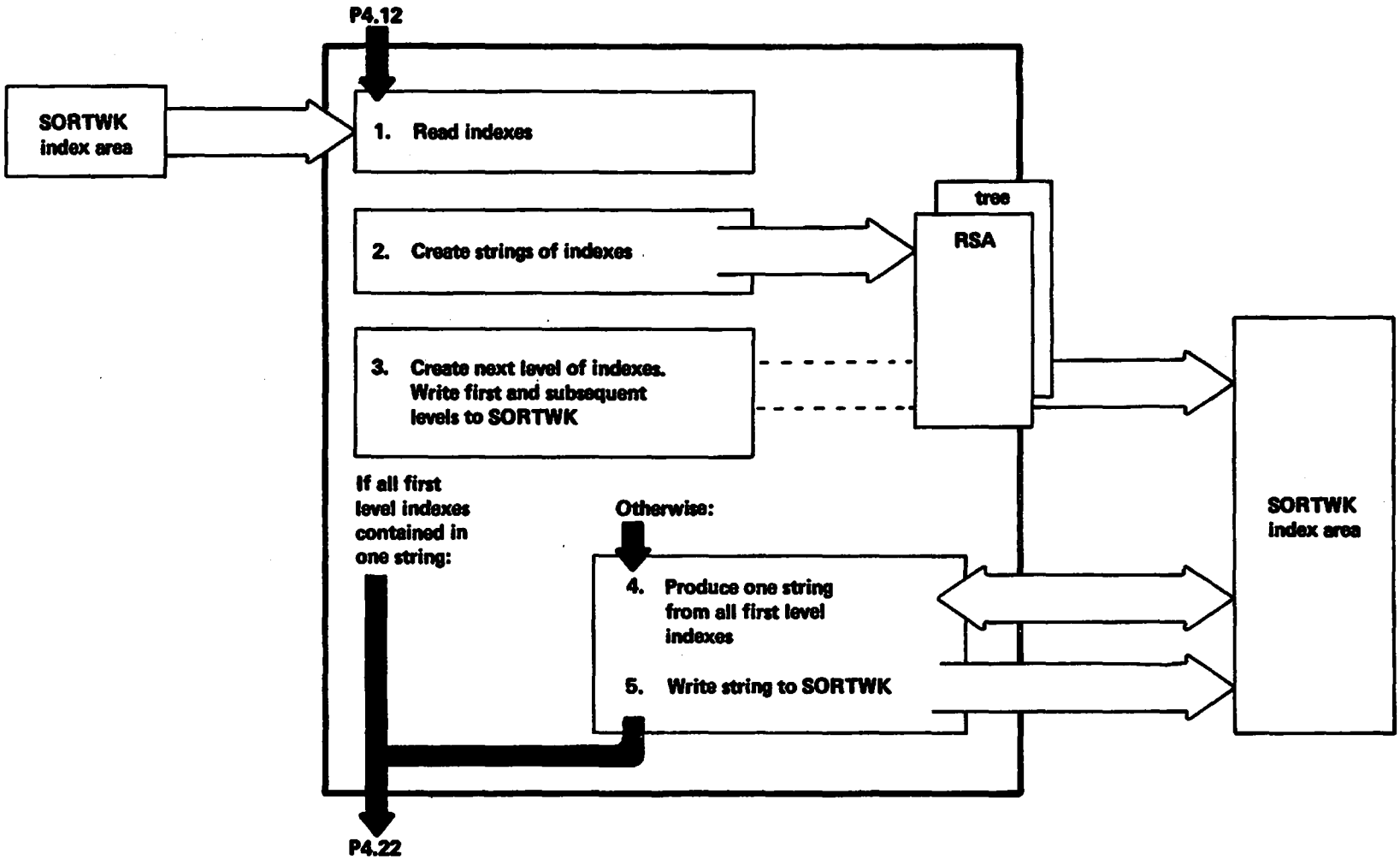
3. Assign each logical block of records to a partition (partitions 0 to n), as described in P4.22. Put the assigned partition number into each index. Partitioning is a form of priming for the following merge step. It enables the merge to read in blocks in the most efficient order, to keep the number of merge passes to a minimum.

4. Sort the indexes (as in point 1), using the partition number as the first control field.

Module
PAR
PAR
PAR

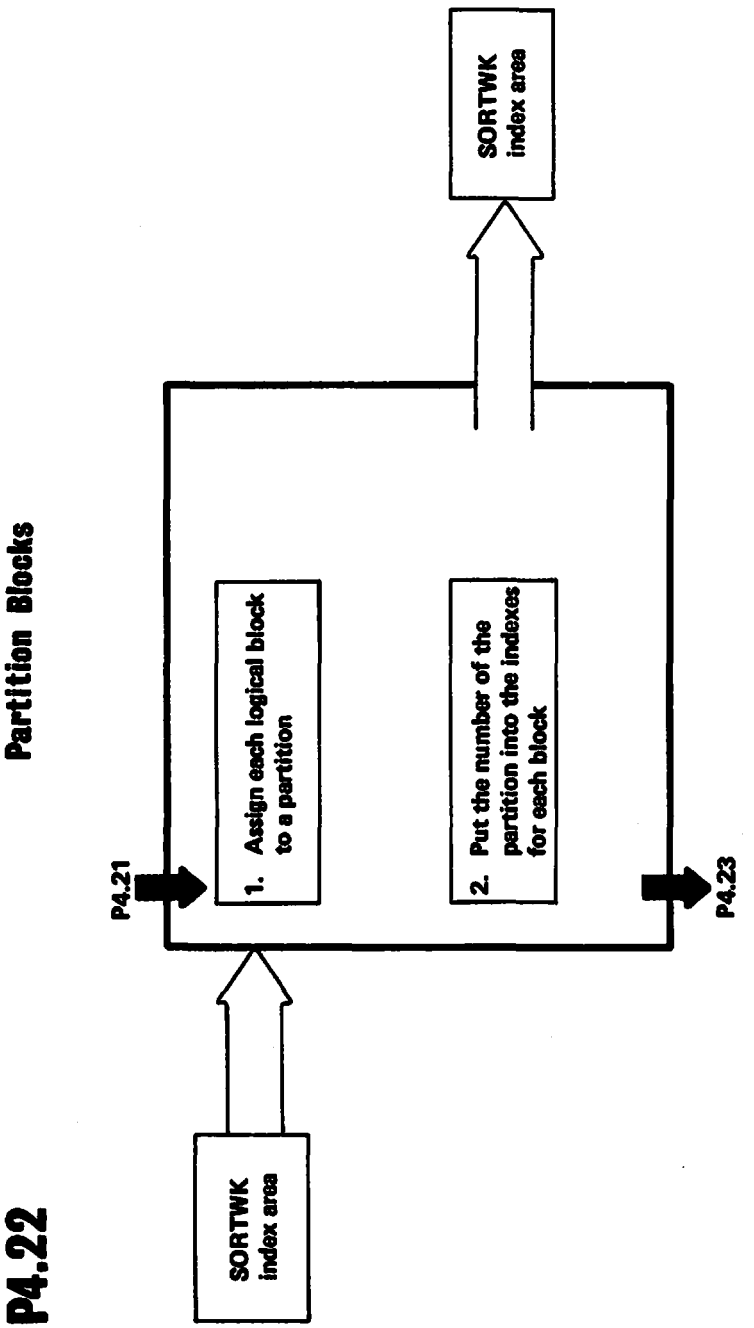
P4.21

Sort Indexes



Extended Description for Diagram P4.21

Peerage only (not used for Vale)	Module
1. Read in the indexes.	PAR
2. Sort the indexes, in the same way as records are sorted in phase 1 (see MO diagram P4.12), using an RSA and tree. The first four fields of the index are used as control fields; the partition number in every index is always zero.	PAR
3. For each logical block of indexes written out, create a new, second-level index and write it. If the second-level indexes cannot all be contained in one string, create a third level, and so on, until a level is reached at which all indexes can be contained in one string.	PAR PAR
If all the sorted first-level indexes are contained in one string, skip points 4 and 5.	
4. Carry out successive merges, until only two levels of index remain. Produce a single string from all the first-level indexes.	PAR PAR
5. Write the string to SORTWK.	PAR
Discard second and higher-level indexes.	



Extended Description for Diagram P4.22

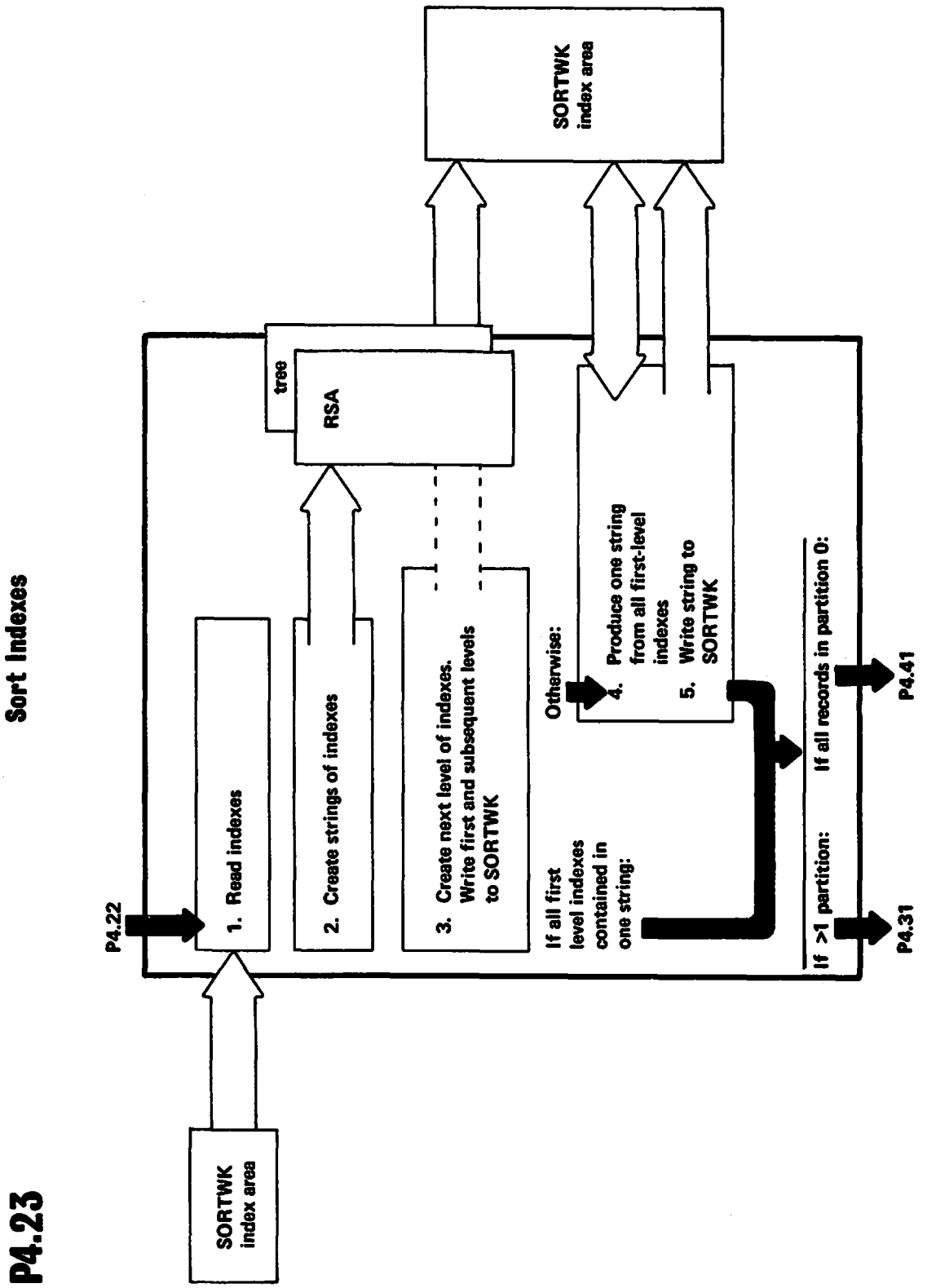
Peerage only (not used for Vale)

1. a Read in the string of indexes created in P4.21.
   To reduce data handling, the indexes are 'mapped': an 8-byte map entry is created for each pair of indexes, using a hash table.

   b Sort the map entries into partitions, in such a way as to maximize the number of logical blocks contained in each successive partition.
   Partition 0 contains M-1 strings, where M is maximum M for phase 3. Each of the other partitions contains M strings, where M is maximum M for phase 2. The last partition contains however many strings remain.

2. Using the hash table, transfer the assigned partition number from each map entry to the corresponding pair of indexes.

Module
PAR
PAR
PAR
PAR



Extended Description for Diagram P4.23

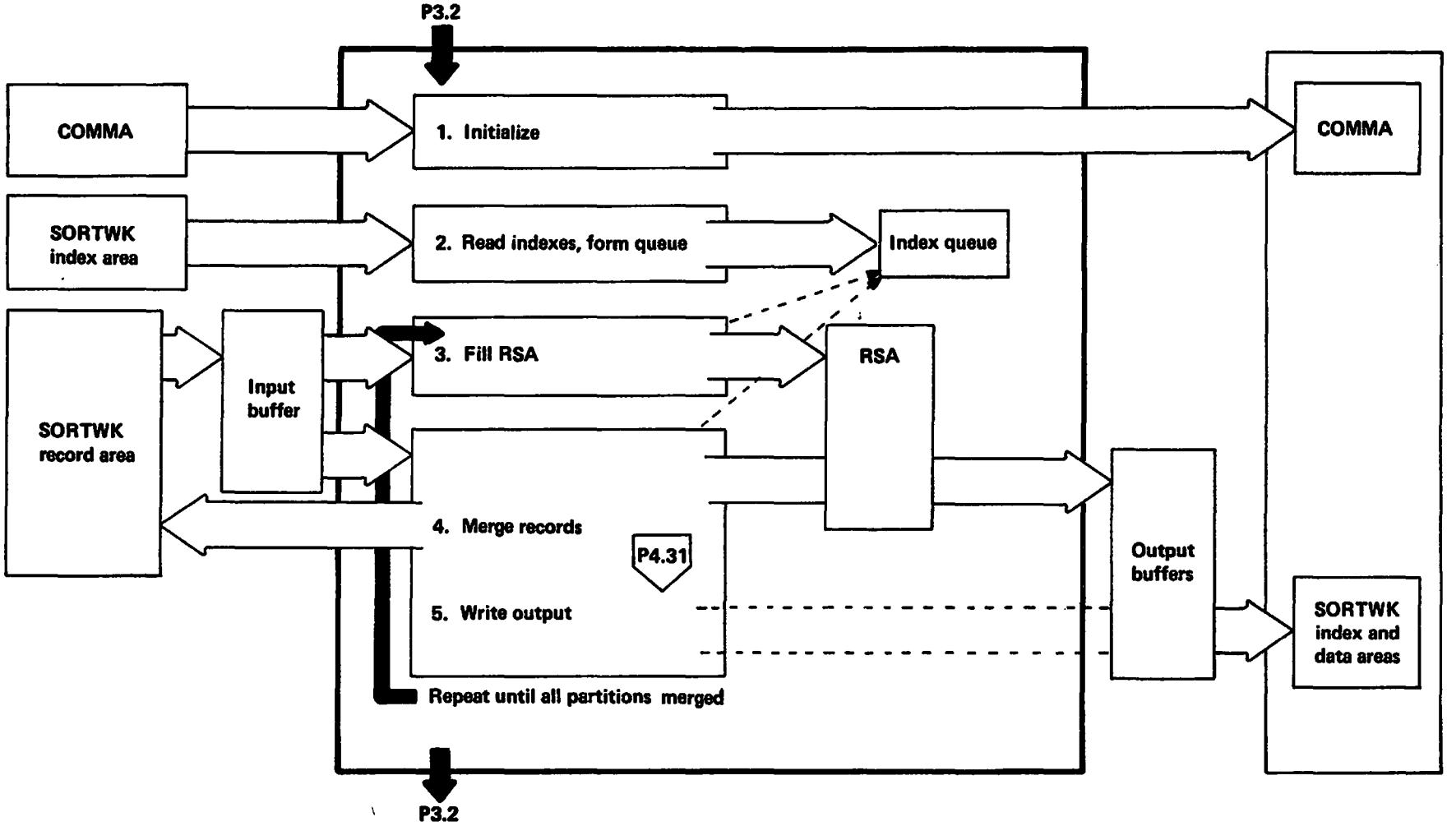
Peerage only (not used by Vale).

The functions are identical to those performed in p4.21, and are carried out by the same segments.

The index sorting carried out here differs from that described in p4.21 in two respects:

- Indexes entering the routine have been updated to contain the correct partition number.
- Entry is from the partitioning routine, and exit is to a merge routine (intermediate or final). In p4.21, entry is from the sort or intermediate merge routine, and exit is to the partitioning routine.

P3.3 Intermediate Merge (Peerage)



Extended Description for Diagram P3.3

An intermediate merge merges strings passed from the partitioning routine into a smaller number of longer strings on SORTWK.

It makes one pass for each partition except partition 0, producing one string from each, and then returns control to the partitioning routine. The partitioning routine continues to call the intermediate merge until the strings can all be contained in partition 0, when it hands control to phase 3 for the final merge.

1. a. Allocate and initialize for SORTWK data sets.

   b. Allocate all remaining main storage to RSA. Chain bins together.

2. a. Read in a block of indexes.

   b. Queue the indexes. Each time an index is removed from the queue, bring in another and merge it into the queue.

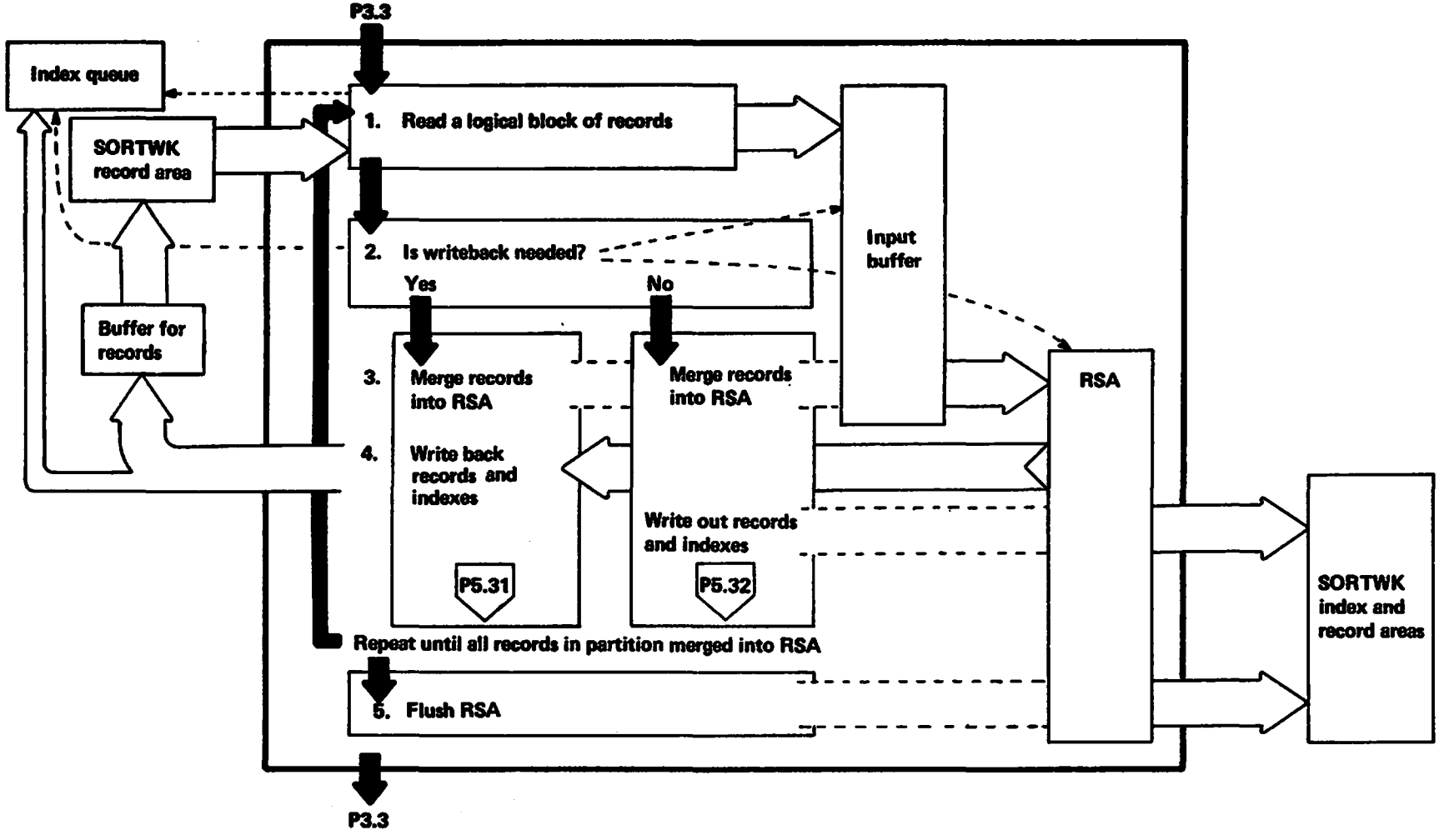
Module
RED
RED
RED
RED

3. Read a logical block of records from the address given in the first index in the queue. Merge the records into the RSA. Remove the index from the queue.
   Repeat until the RSA is full.

4,5 Read a block of records from SORTWK to an input buffer.
   If necessary, read a block back from the RSA to make room for the incoming block (the criterion for writeback is described in MO diagram P4.31).
   Otherwise write a block out from the RSA, and create indexes for it.
   Merge the incoming block into the RSA.
   Continue until the entire partition has been merged and indexed into one string on SORTWK.

   Carry out points 3-5 for each partition except Partition 0.

P4.31 Merge

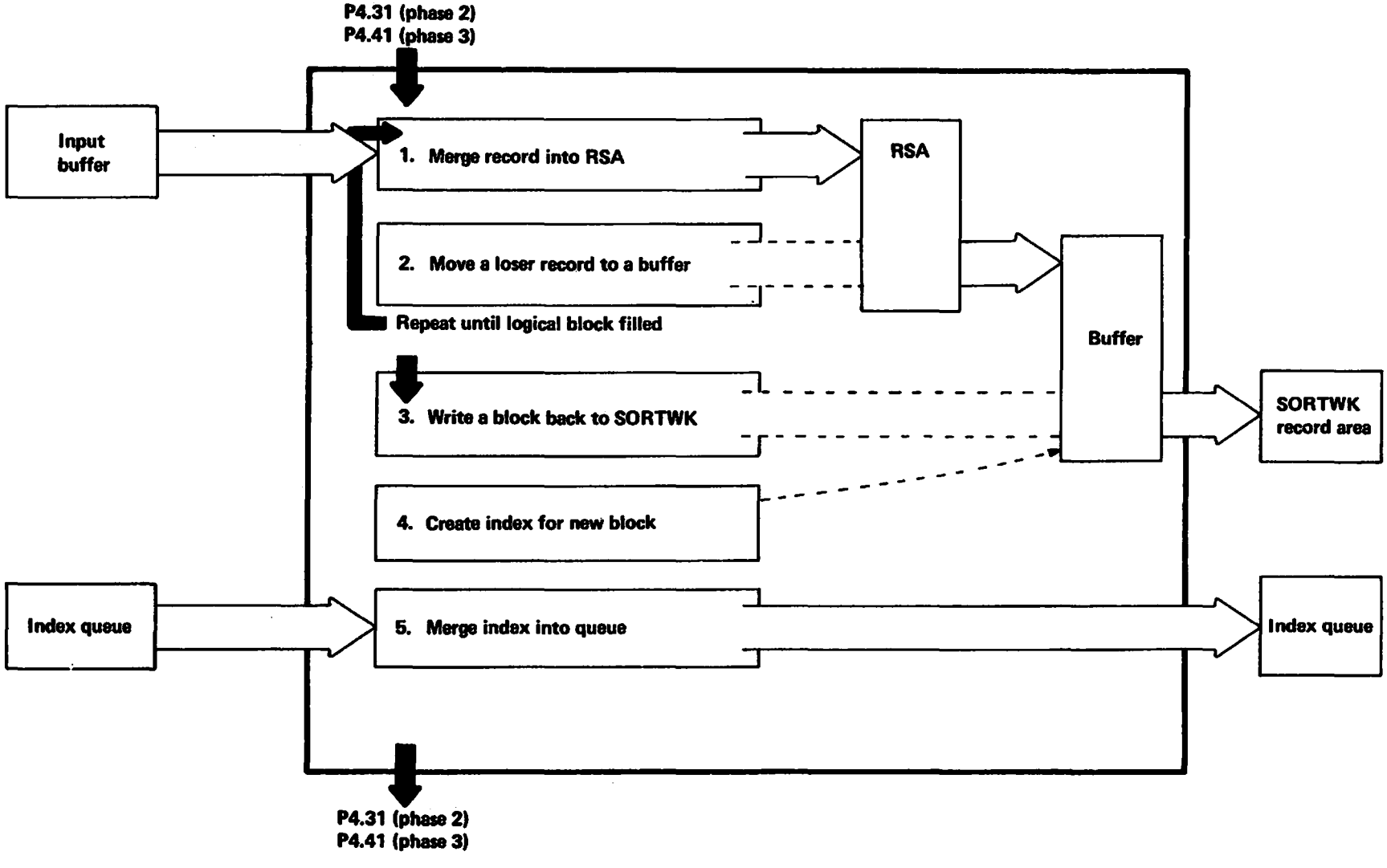


Extended Description for Diagram P4.31

	Module
1. Read into the input buffer the logical block of records pointed to by the first index in the index queue.	RED
2. Find out whether enough records can be written out of the RSA to make room for the incoming block. The space which can be made available is equal to the number of empty bins, plus the number of records (in the RSA and in the incoming block) which 'win' over the first record in the next block that will be read in (pointed to by the next index in the queue). If this space is smaller than the number of records in the incoming block, writeback is needed.	RED
3. Merge the incoming block into the RSA, each record forcing out one to a buffer.	RED RED
4. Create an index for the new block and move the block to SORTWK.	RED
If writeback is required, add the index to the index queue; otherwise, write out the index to SORTWK.	RED
5. When there is no more input in this partition, flush out the records remaining in the RSA. The partition is now in one string on SORTWK.	RED

P5.31

Writeback



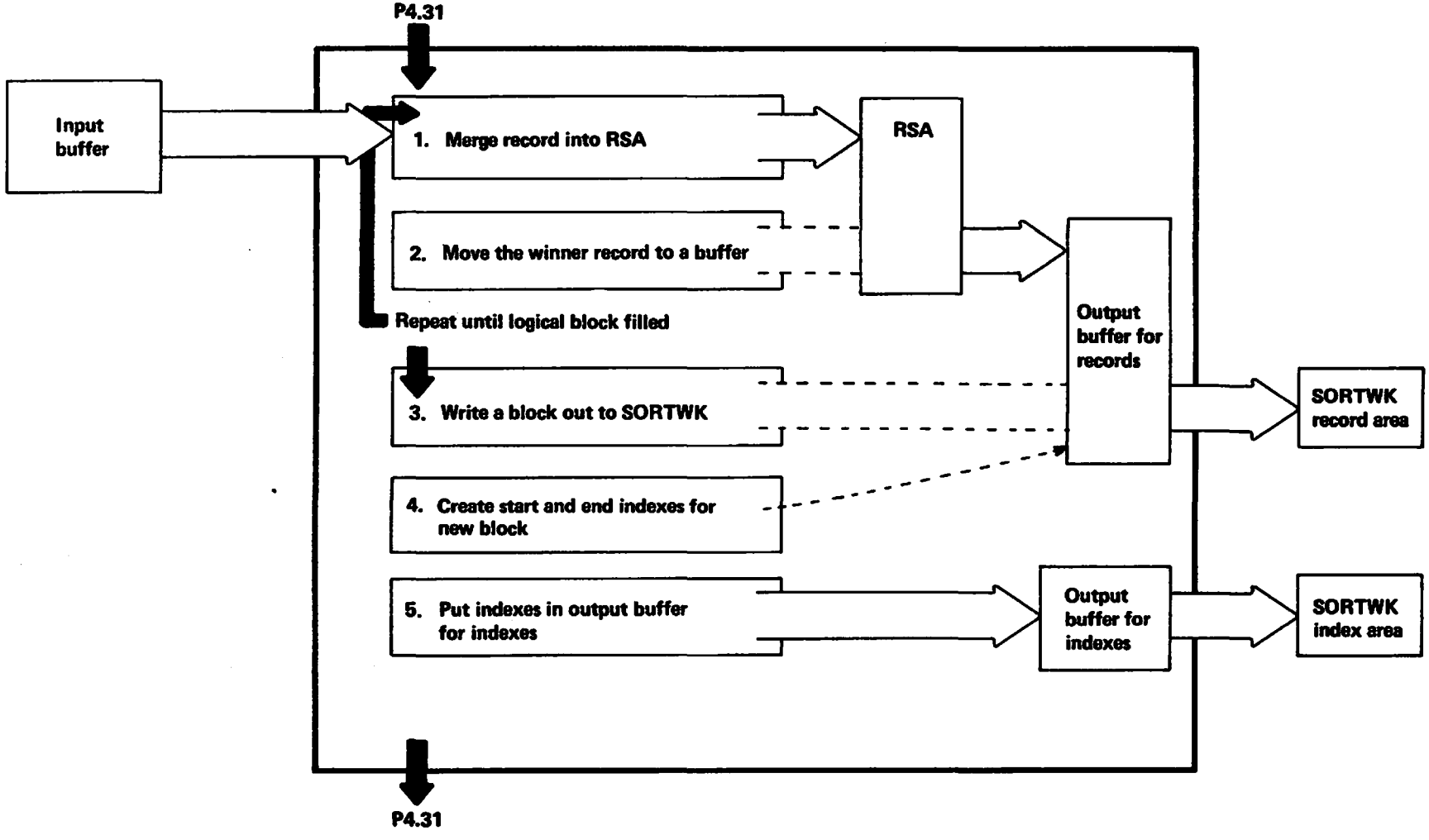
Extended Description for Diagram P5.31

	Module
1. Move the first record from the input buffer into the first vacant bin in the RSA, and chain into its correct position.	***
2. Move a record from the losing half of the RSA out to a buffer.	***
3. When a logical block is complete, move it to SORTWK.	***
4. Create an index for the new logical block.	***
5. Merge the new index into the index queue (from which blocks of records will be selected for reading in to the current merge pass).	***

*** = RED in phase 2, LIM/LIV in phase 3.

P5.32

Write



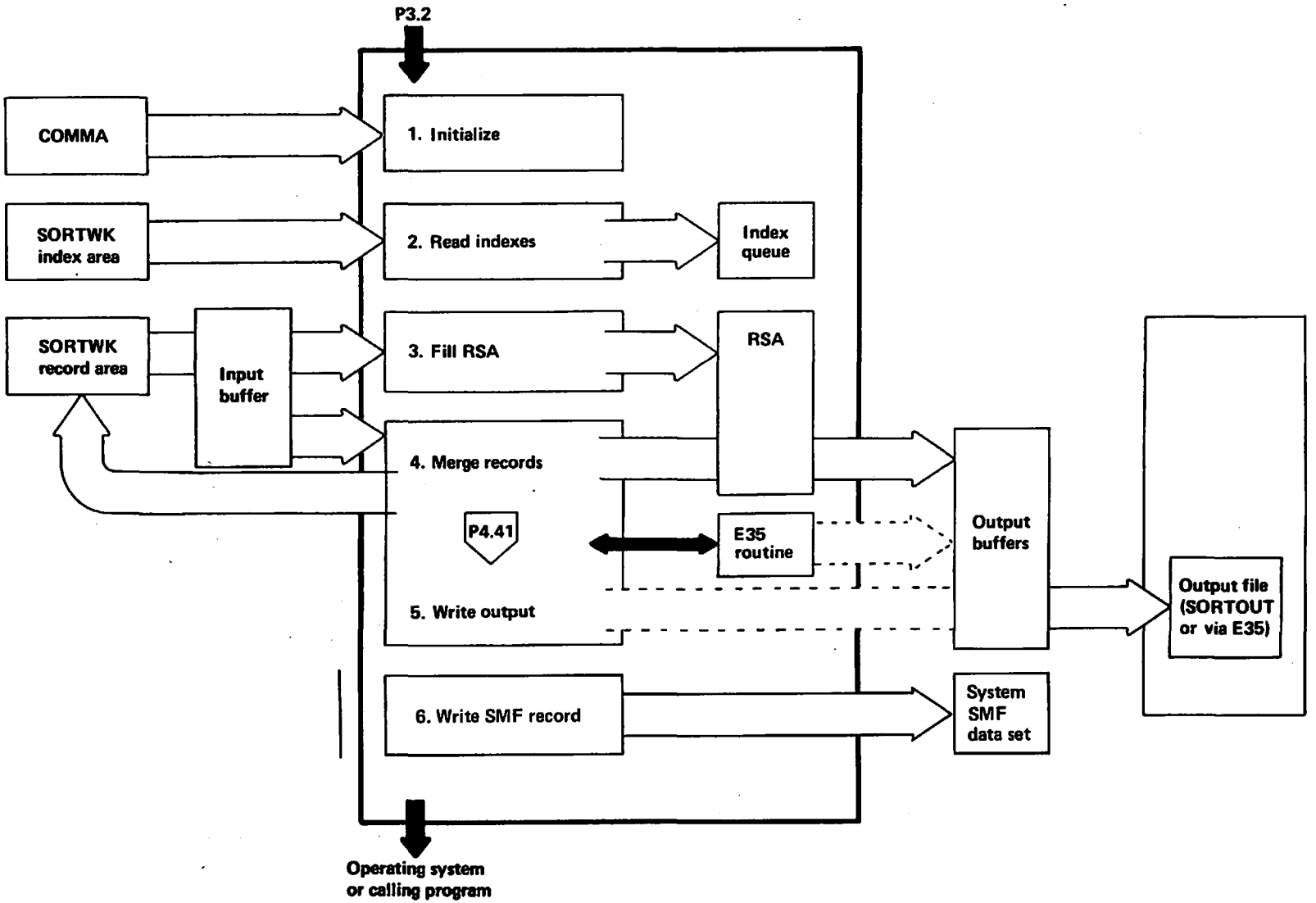
Extended Description for Diagram P5.32

The difference between this function and P5.31 (writeback) is in the handling of the indexes: here, they are written out to the area passed to the partitioning routine; in writeback, they are added to the index queue, so that the block to which they refer will be read in again in the current merge pass.

	Module
1. Move the first record in the input buffer into the first vacant bin in the RSA. Chain it into its correct position.	RED
2. Move the winner record from the RSA to the output buffer.	RED
3. When a logical block is complete, move it to SORTWK.	RED
4. Create indexes for the new logical block.	RED
5. Move the new indexes to the index output buffer; when the buffer is full, write it to SORTWK.	RED

P3.4

Final Merge (Peerage)



Extended Description for Diagram P3.4

The differences between the final merge and the intermediate merge (P3.3) are that: input consists of only one partition, partition 0; no new indexes are created; and output is to SORTOUT (or equivalent), not to SORTWK.

1. a. Allocate and initialize for SORTOUT and SORTWK data sets. Open the SORTOUT data set if it is not VSAM.

   b. Allocate all remaining main storage to the RSA. Chain bins together.

2. a. Read in a block of indexes.

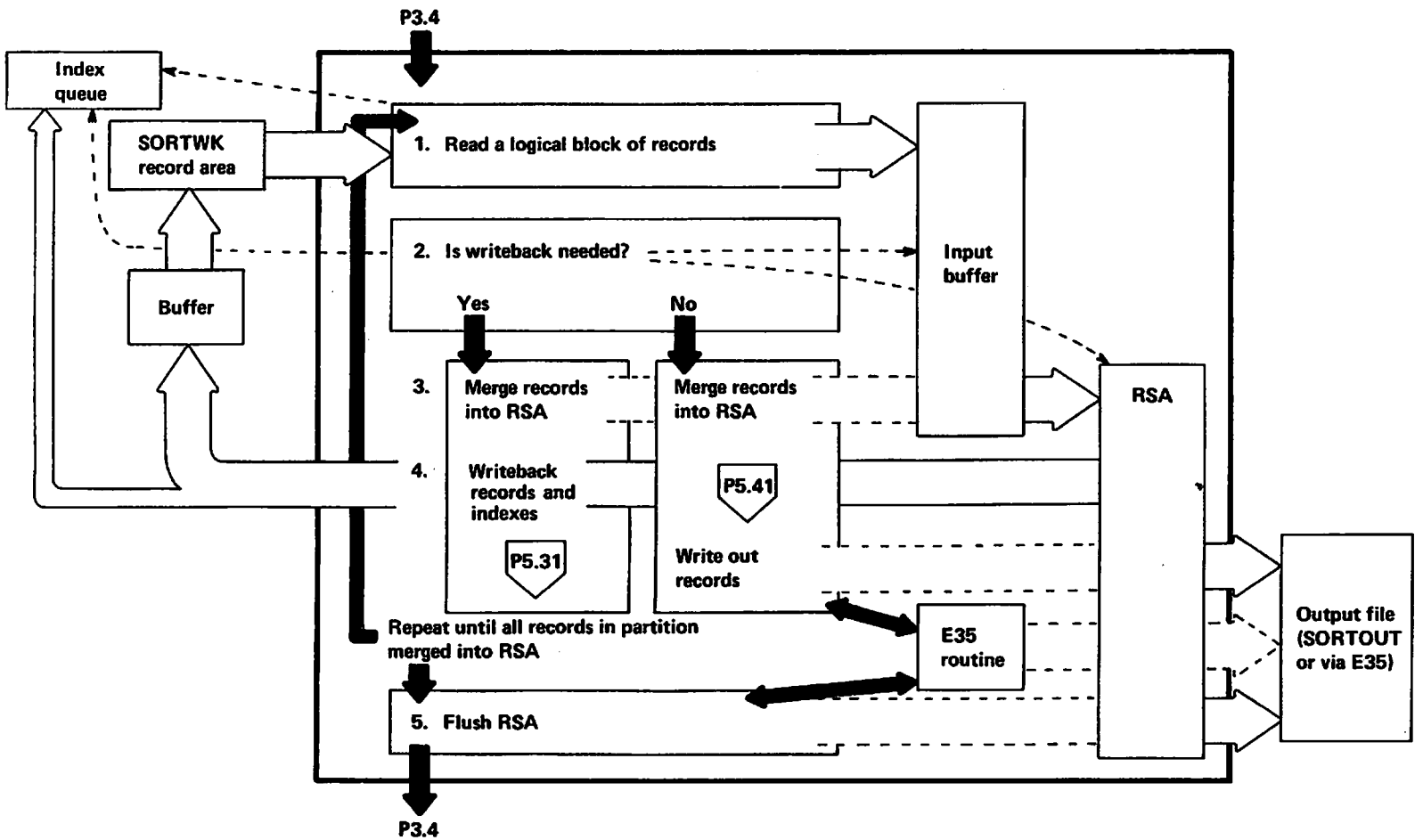
   b. Queue the indexes. Each time an index is removed from the queue, bring in another and merge it into the queue.

3. Read a logical block of records from the address given in the first index in the queue. Merge the records into the RSA. Remove the index from the queue.
   Repeat until the RSA is full.

Module	
LIM/LIV	4,5 Read a block of records from SORTWK to an input buffer. If necessary, read a block back from the RSA to make room for the incoming block (the criterion for writeback is described in MO diagram P4.31). Otherwise write a block out from the RSA, taking E35 after each record is moved out. Merge the incoming block into the RSA. Continue until the entire partition has been merged into one string on SORTOUT.
LIM/LIV	
LIM/LIV	6. Write SMF record type 16 to system SMF data set, if requested.
LIM/LIV	
LIM/LIV	

P4.41

Merge



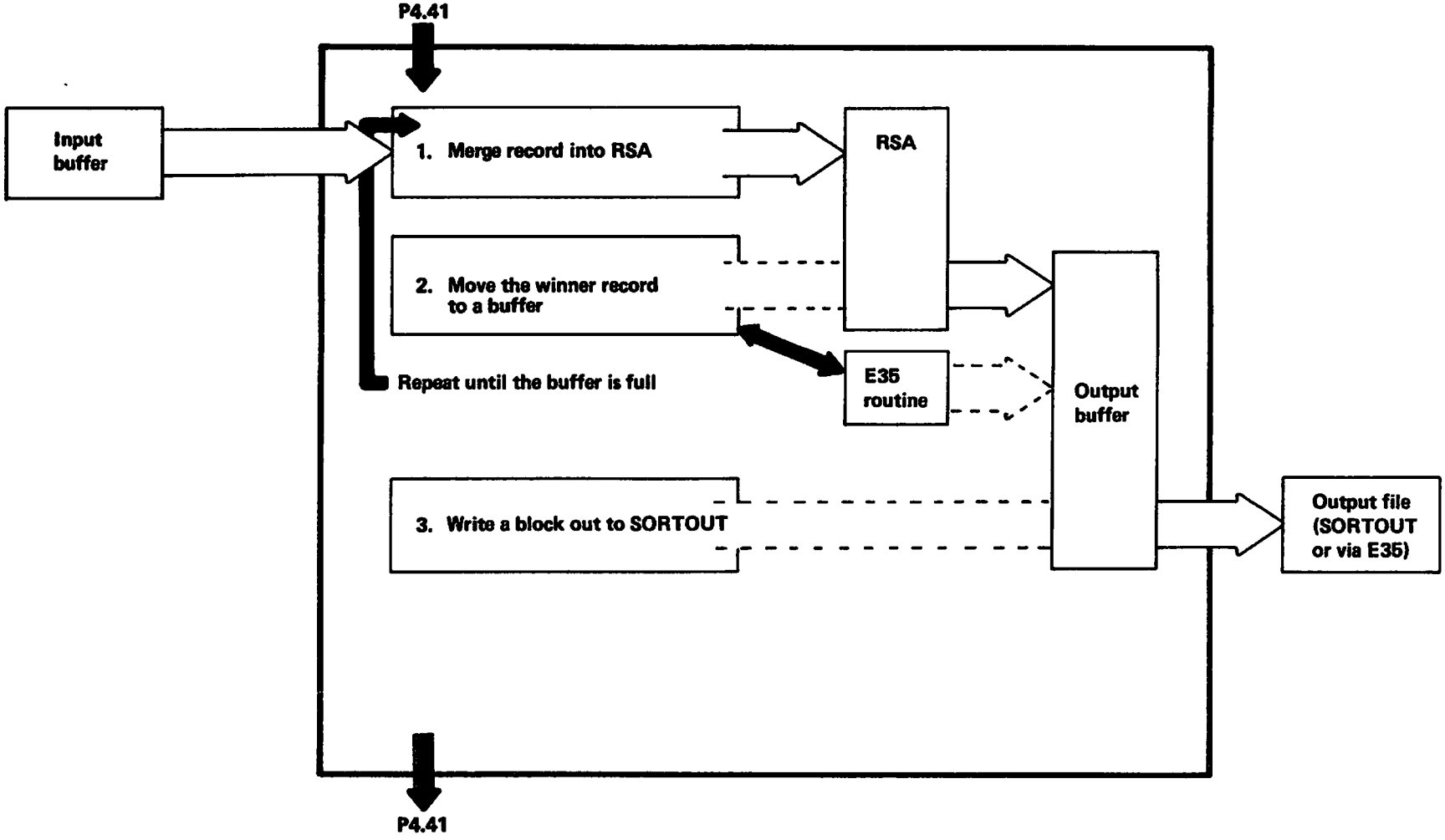
Extended Description for Diagram P4.41

	Module
1. Read into the input buffer the logical block of records pointed to by the first index in the index queue.	LIM/LIV
2. Find out whether enough records can be written out of the RSA to make room for the incoming block. The space which can be made available is equal to the number of empty bins, plus the number of records (in the RSA and in the incoming block) that 'win' over the first record in the next block which will be read in (pointed to by the next index in the queue). If this space is smaller than the number of records in the incoming block, writeback is needed.	LIM/LIV
3. Merge the incoming block into the RSA, each record forcing out one to a buffer. Take exit E35.	LIM/LIV* LIM/LIV
4. Move the block to SORTWK. If writeback is required, Create an index for the new block, and add the index to the index queue.	LIM/LIV
5. When there is no more input, flush out the records remaining in the RSA. The data set is now in one string on SORTOUT.	LIM/LIV/XCP

*Code generated outside LIM's area; address kept in COMMA at CMERGE.

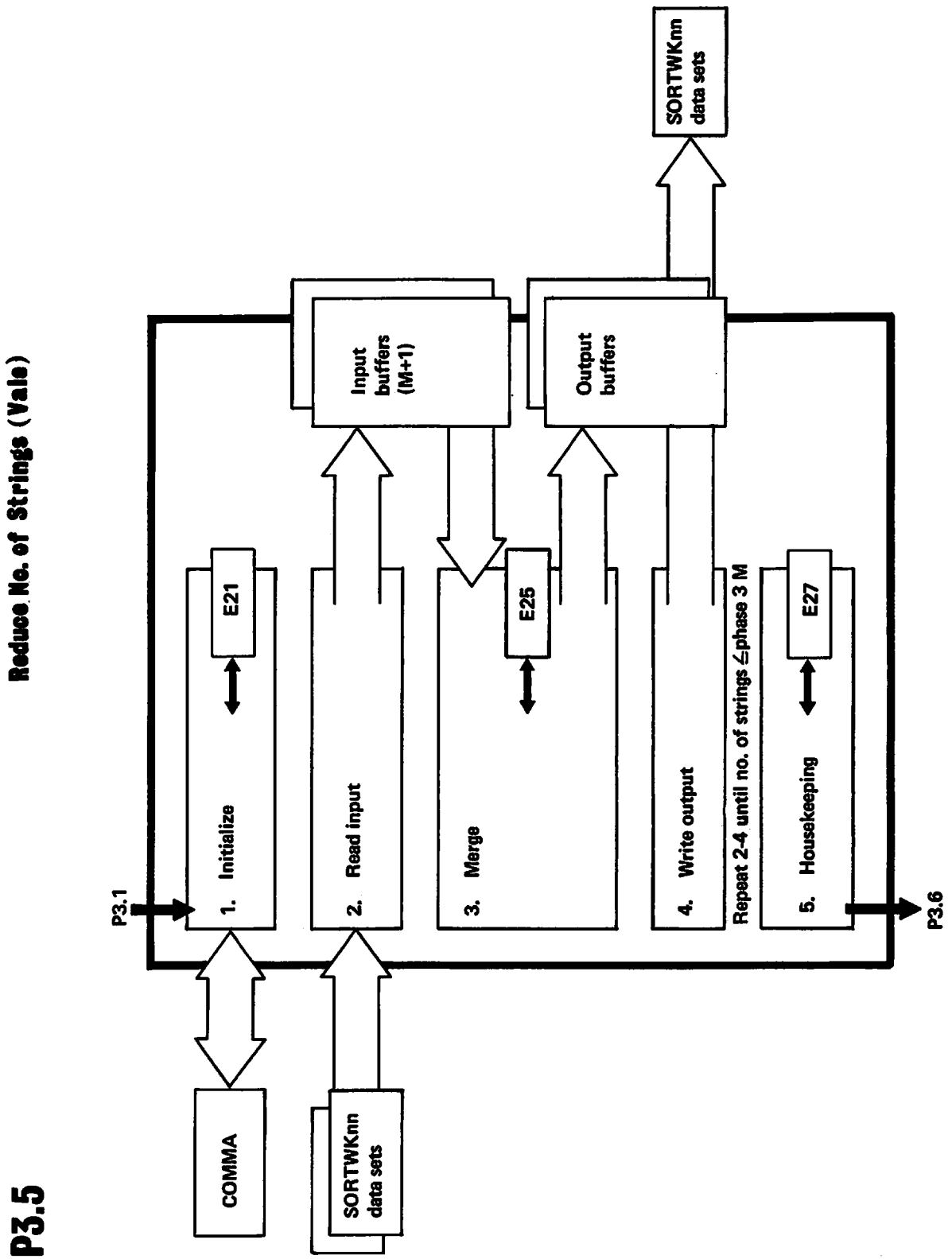
P5.41

Write Out



Extended Description for Diagram P5.41

	Module
1. Move the first record in the input buffer into the first vacant bin in the RSA. Chain it into its correct position.	LIM/LIV/VIM
2. Take E35 if specified. Move the winner record from the RSA to the output buffer.	LIM/LIV/VIM
3. When the buffer is full, move it to SORTOUT.	LIM/LIV/ VIM/XCP



Extended Description for Diagram P3.5

These functions are carried out by module VED; if exit E25 has been specified an additional module, VEE, is used to handle the interface with the E25 routine.

1. a. Allocate buffers: for input, phase 2 merge order plus one; for output, two.

   b. Initialize the phase work area.

   c. Load and call E21 and load E25, if specified.

2. At the beginning of a merge pass:

   a. Fill the input buffers.

   b. Make a simple tree, with one pointer to the first record in each buffer, and the pointers (physically) in the desired output order.

   Subsequently:

   c. Keep the pointers updated to point to the current record in each buffer.

   d. Keep the tree order so that the pointers are in the desired output order.

   e. Replenish the input buffers when necessary, using the extra input buffer for 'smart lookahead'.

3. a. If E25 is <u>not</u> activated:

      Move the record pointed to by the first pointer in the tree from its input buffer to the output buffer.

   b. If E25 <u>is</u> activated:

      Move the record pointed to by the first pointer in the tree from its input buffer to the E25 work area. Each new record moved to the work area forces the preceding record out to the output buffer. Take E25.

4. When the output buffer is full, write it to a SORTWK data set and chain it into the correct place in the string.

Repeat 2c-4 until input is exhausted. If the number of output strings is now less than or equal to maximum phase 3 M, continue to point 5. Otherwise, return to 2a for another merge pass.

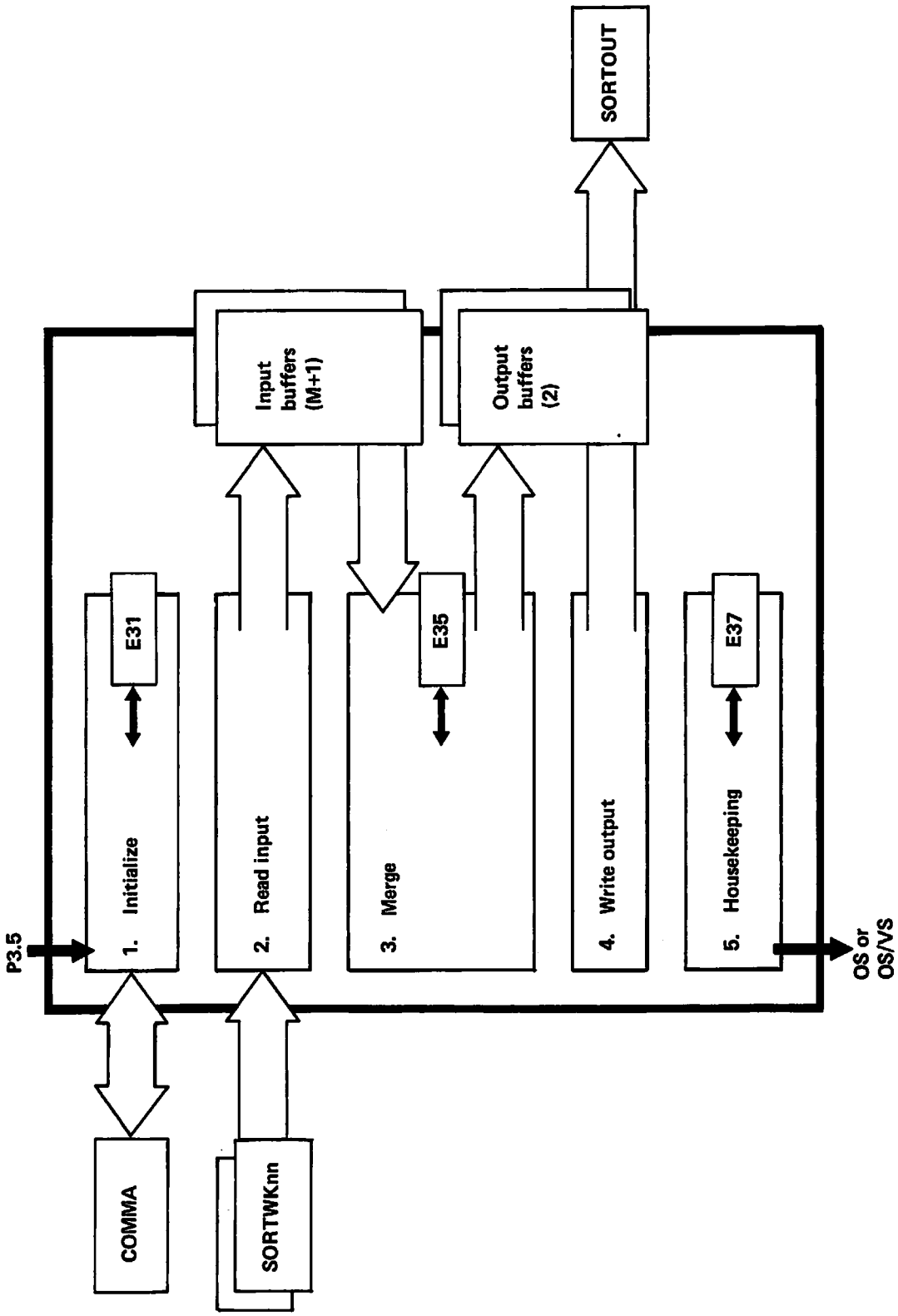
5. a. Load and call E27 if specified.

   b. E25 housekeeping.

   c. Free temporarily allocated areas.

P3.6

Final Merge (Vale)



Extended Description for Diagram P3.6

1. a. Allocate buffers: for input, phase 3 M plus one; for output, the number determined in phase 0.

   b. Initialize the phase work area.

   c. Load and call E31 if specified.

2. At the beginning of the phase:

   a. Fill the input buffers.

   b. Make a simple tree, with one pointer to the first record in each buffer, and the pointers (physically) in the desired output order.

   Subsequently:

   c. Keep the pointers updated to point to the current record in each input buffer.

   d. Keep the tree ordered so that the pointers are in the desired output order.

   e. Replenish the input buffers when necessary, using the extra input buffer for 'smart lookahead'.

3. If E35 has been specified:

   a. Move the record pointed to by the first pointer in the tree from its input buffer to a work area, restoring its original format.

   b. Call E35.

   c. On return, move the record passed by E35 to an output buffer.

   Otherwise, move the record pointed to by the first pointer in the tree from its input buffer to an output buffer, restoring its original format.

4. When the output buffer is full, write it to the SORTOUT data set.

Repeat 2c-4 until input is exhausted.

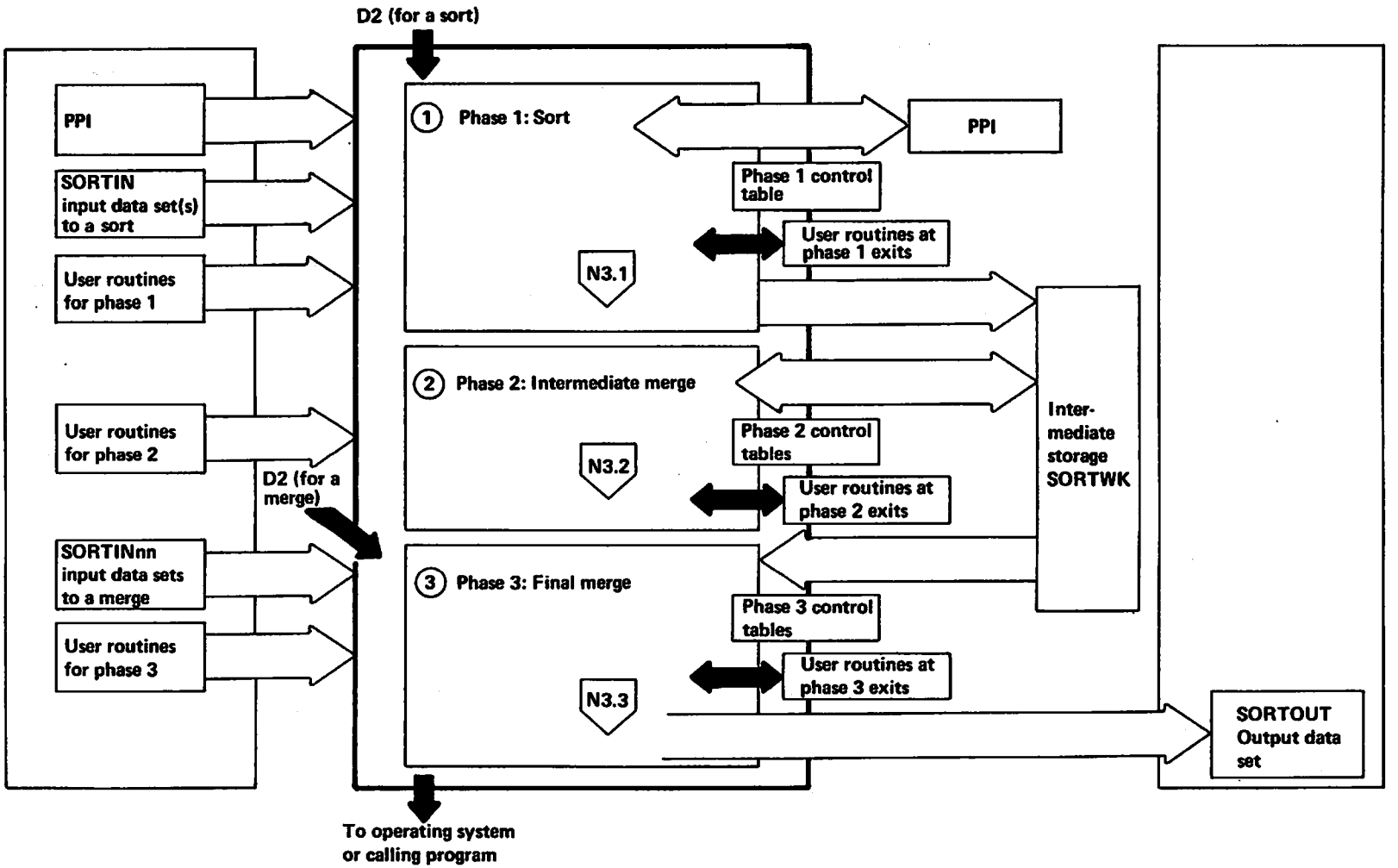
5. a. Load and call E37 if specified.

   b. Delete all exit routines.

   c. Free temporarily allocated areas.

   d. Write SMF record, if requested.

Conventional Sort/Merge



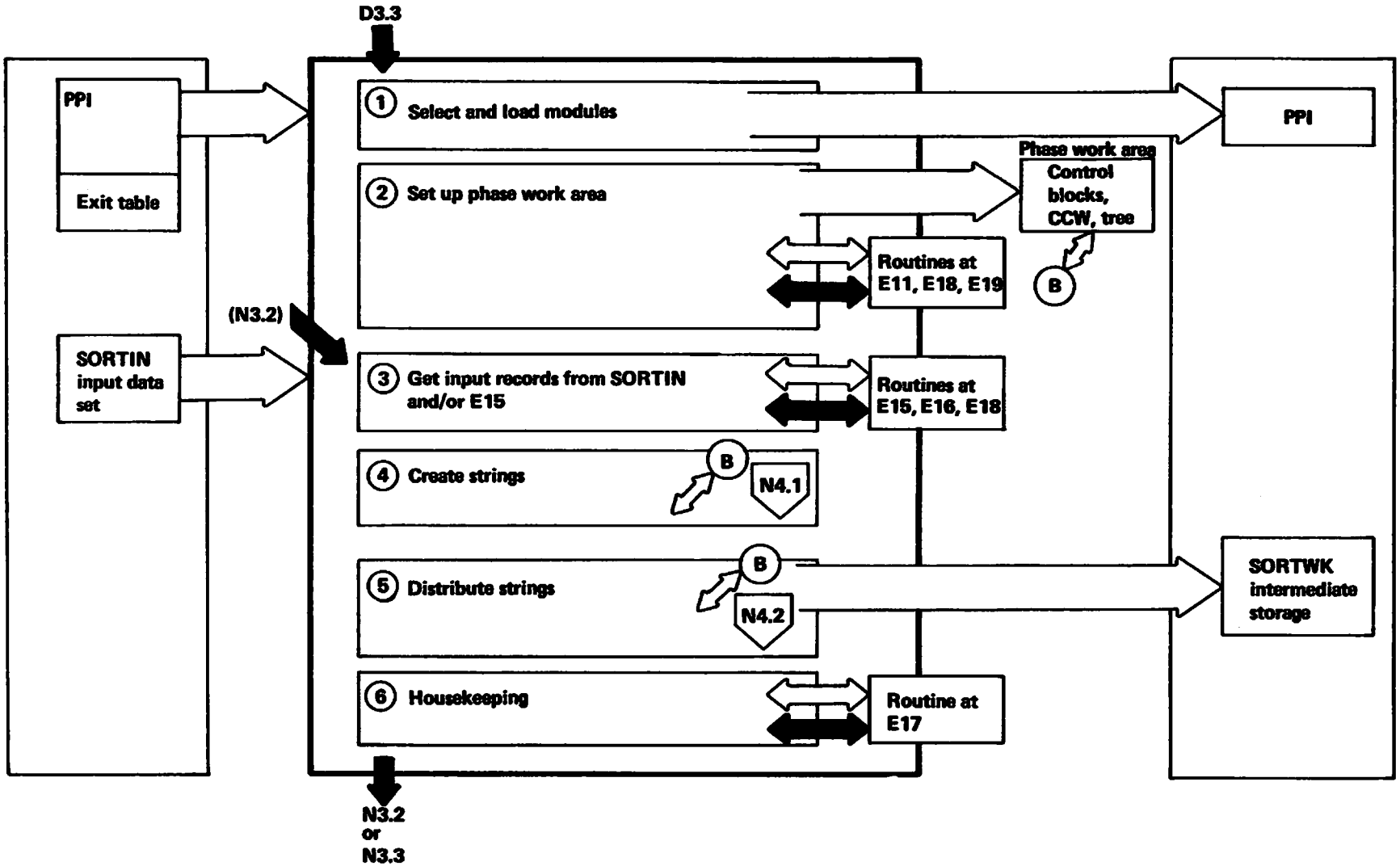
Extended Description for Diagram N2

1. Initialize for phase 1. Take E11, E18, E19.
   Get records from SORTIN. Take E15, E16, E18.
   Create strings, and distribute them on SORTWK. The sorting technique used is the replacement selection technique.
   Take E17.

2. Initialize for phase 2. Take E21, E28, E29.
   Read strings of records from SORTWK.
   Take E28.
   Merge the strings into longer strings. Take E61, E25.
   Write out the new strings to SORTWK.
   Take E29, E27.

3. Initialize for phase 3. Take E31, E38 and E39.
   Read strings of records from SORTWK.
   Take E32, E38.
   Merge strings into output buffer.
   Take E61.
   Write out single string to SORTOUT.
   Take E35,E39,E37.
   Perform final housekeeping.

N3.1 Sort (Phase 1)



Extended Description for Diagram N3.1

1. The selection of modules is made by ICERC6 using information from PPISW1 combined with the module table TBLPH1RN (which is defined in ICERC6).
   Information about user routines not link edited by the program is taken from the Exit Name Table (. PPIAPGC).
   ICERC9 a) loads all running modules and user routines, then b) loads, executes and deletes the assignment modules one by one.

2. Take exit 11. Acquire main storage (through GETMAIN) for the phase work area, buffers, and RSA.
   Initialize the tree in the phase work area.
   Generate the move routine.
   Generate control blocks (DCB, IOB, ECB) and CCWs in the phase work area.
   Take exits E18 and E19.
   Define the RSA bin structure, and enter the relevant information in the PPI.
   Open the SORTIN and SORTWK data sets.
   Records from SORTIN are skipped if the SKIPREC option is present.
   Code in the running modules is initialized by the respective assignment modules.

If the CRCX or OSCL technique is being used some further functions are performed here which otherwise belong to phase 2; see MO diagram N3.2 for details.

3. Read SORTIN, using QSAM or VSAM.
   Locate mode is normally used, but move mode is used for spanned records. Each deblocked variable-length record has its length checked. VSAM variable-length records are supplied with a record descriptor word (4 bytes at the front of the record).
   Take exit E15 (record may be altered or deleted or a new record inserted).
   If calculated capacity is exceeded, take exit E16.

4. Fill the RSA, updating the tree for each record.
   Create strings by moving records from the RSA to a buffer, using the tree to select each record to be moved.

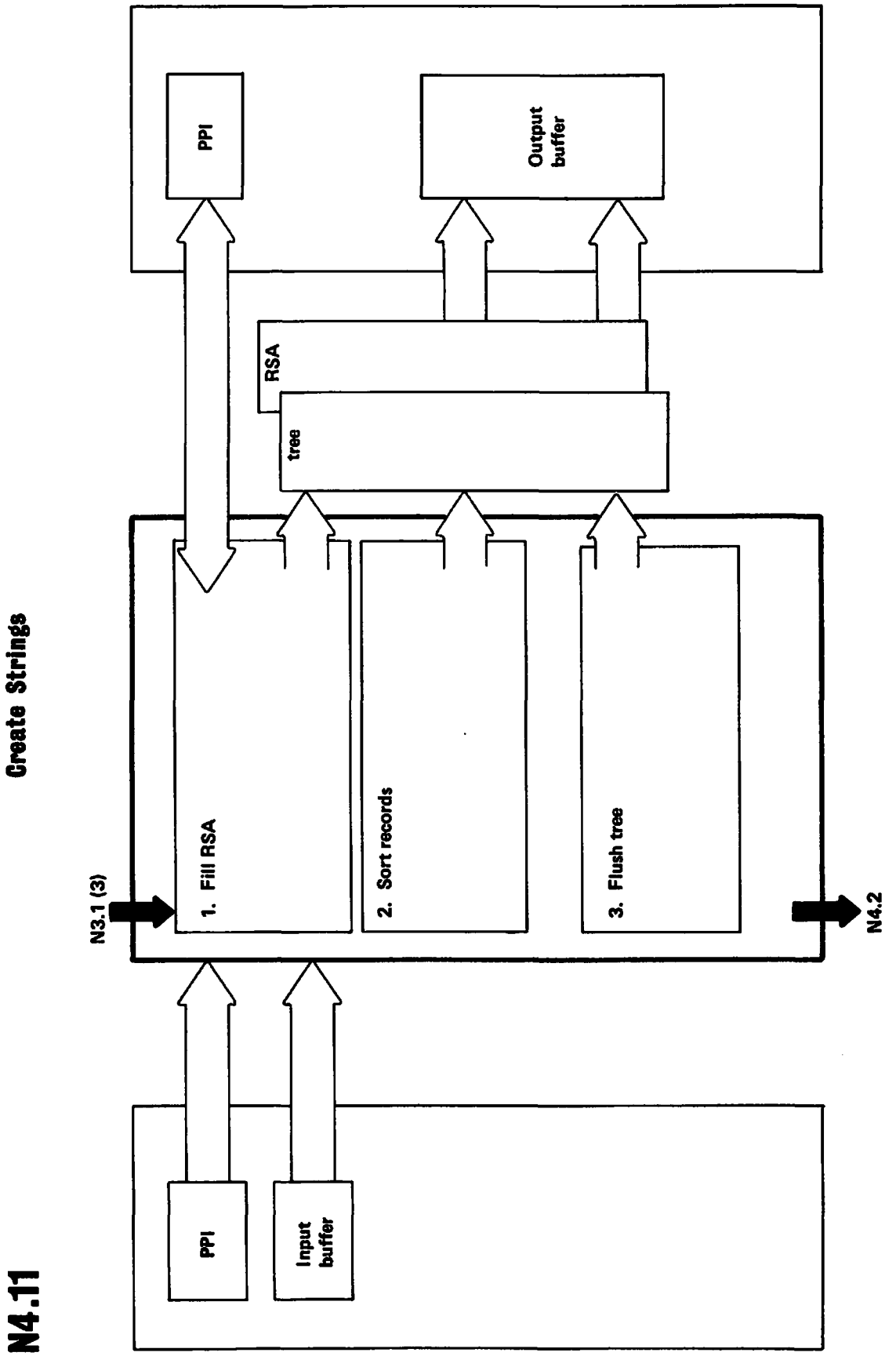
5. Write output, using the EXCP macro instruction.
   The sequence in which strings are written to the various SORTWK data sets is determined by the sequence distribution technique used.

6. If exact file size was specified, check the record counts.
   Take exit E17.
   Close SORTWK data sets.
   Free main storage and delete modules.

Module Table for Diagram N3.1

Function	Conditions				Mod
1a Select modules	OSCL,CRCX		phase 1		RC6
			phase 2		RC7
	Other				RC6
1b Load modules	Always				RC9
2a Allocate storage	CRCX,OSCL				APL
	Otherwise				APG
2b Build blocks and tables	POLY,BALN(t)				AGA
	OSCL				AGN
	BALN(d)				AGI
	CRCX				9GN
3a Deblock	Fix	Exit	CRCX,OSCL		RDR
			Other		RDD
		No E	256+	CRCX,OSCL	RDQ
				Other	RDC
			<256	CRCX,OSCL	RDR
				Other	RDB
	Var	Exit	CRCX,OSCL		RDS
			Other		RDE
		No E	CRCX,OSCL		RDT
			Other		RDG
3b VSAM read (additional to deblock)					RGV
6 Housekeeping (For CRCX & OSCL, carried out in ph2)	Other than CRCX,OSCL				RPC

This page intentionally left blank



Extended Description for Diagram N4.11

1. a. Move a record from the input buffer to the RSA.

   b. Update the tree.

2. a. Move the winner from the RSA to the output buffer.

   b. Move in the next record from the input buffer.

   c. Update the tree.
   Repeat until there are no more input records; unless the OSCL or CRCX technique is being used, in which case repeat until M-1 strings have been moved to the output buffer.

3. Move remaining records from the RSA to the output buffer.

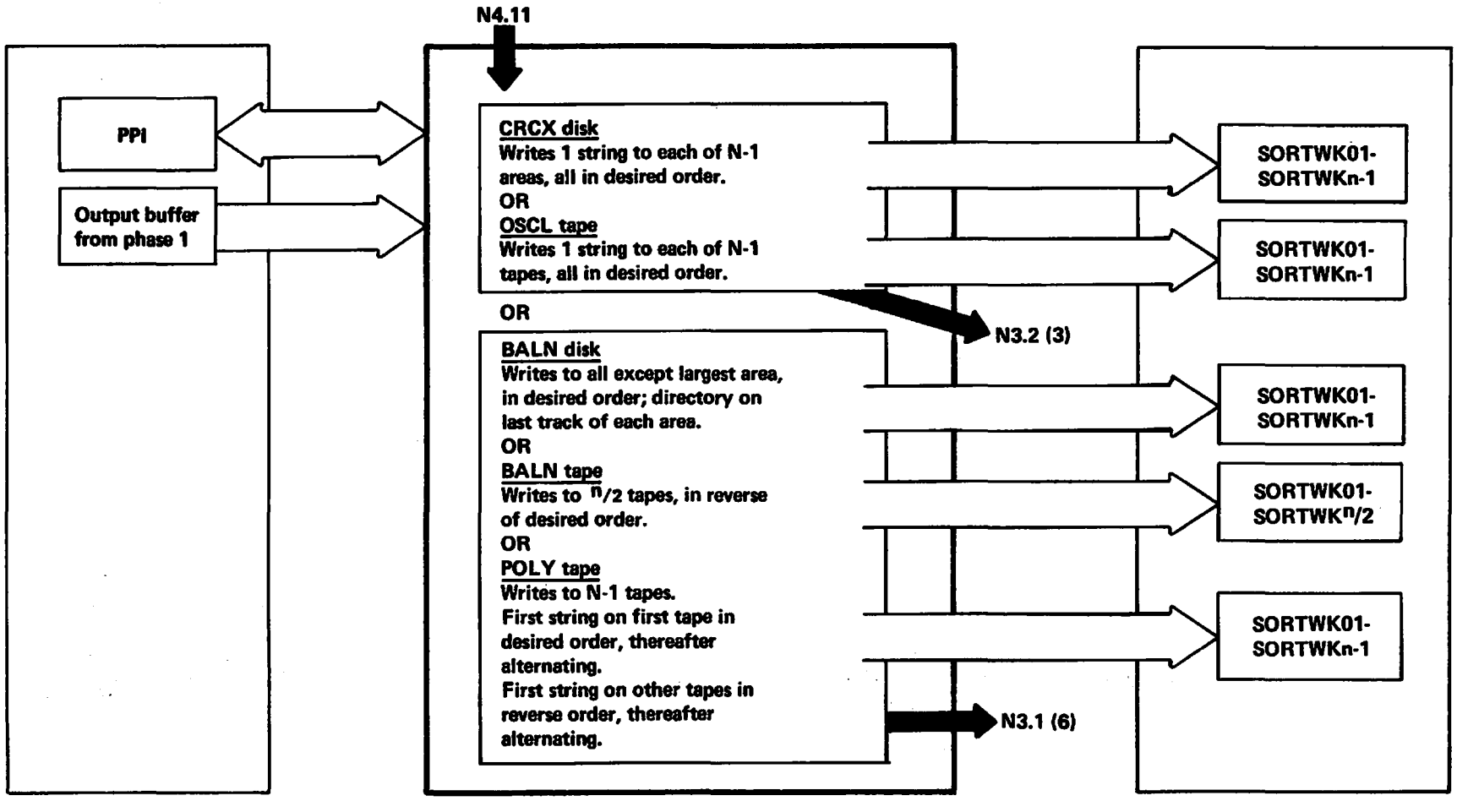
Module Table for Diagram N4.11

Function	Conditions			Mod
1a Move	Fix	<256, no E		- [1]
		Other		ABS[2]
	Var			RBF
1b Update tree	Fix	Mult.	POLY	ROE
			Other	ROA
		Single	POLY	ROF
			Other	ROB
	Var	Mult.	POLY	ROG
			Other	ROC
		Single	POLY	ROH
			Other	ROD
2a Block	Fix	256+	CRCX,OSCL	RBZ
			Other	RBC
		<256	CRCX,OSCL	RBY
			Other	RBB
	Var		CRCX,OSCL	RBA
			Other	RBE

[1]No special move routine needed--the deblock routine used.
[2]This module generates a move routine which is kept in main storage (pointed to by PPIBDSVA+4); if the EQUALS option is in operation, an additional routine is also generated (↑PPIMOVEQ).

N4.12

Distribute Strings



Extended Description for Diagram N4.12

CRCX disk
One string is written to each of N-1 areas. If a physical area is too small to contain the whole string one or more tracks are borrowed from another physical area. These together make up the logical area. This can be done because each block within the string contains the disk address of the following block; the last block in a string contains its own address. Starting addresses of strings (↑ KTABLEL) and free track list (↑ KFRTRKLT) are kept in the phase work area.

BALN disk
Strings are written to N-1 areas in order of size, starting with the smallest. Eight strings are written to each area at a time, completely filling one directory block. If a complete string does not fit into the remainder of an area it is divided into substrings, the first completely filling the area.
The directory contains the starting address of the string, and the remaining blocks follow consecutively. The last block of the string contains a special EOS indicator.

BALN tape
Strings are written to N/2 tapes, one at a time. If a string does not fit on a tape it is divided into substrings.

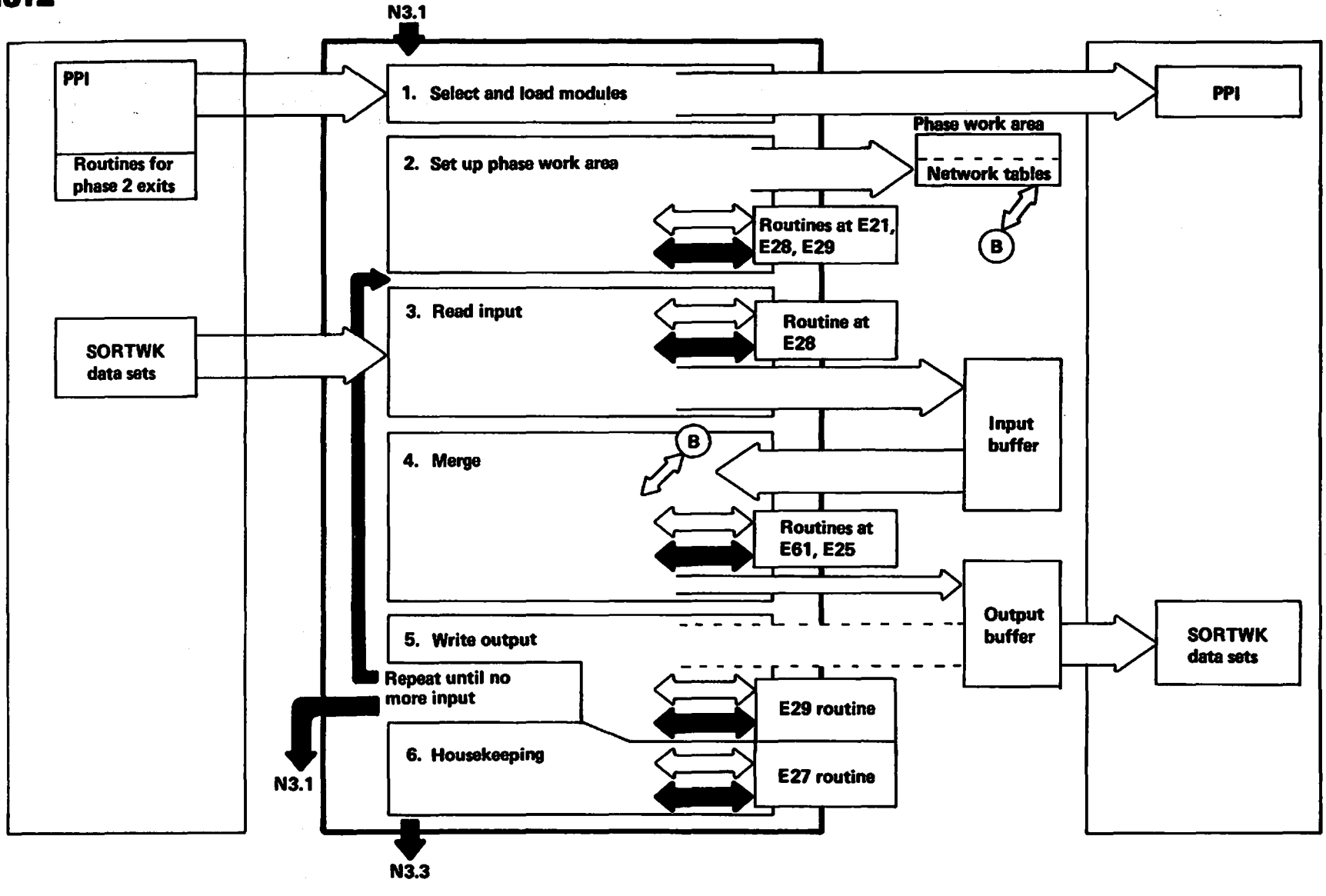
OSCL tape
One string is written to each of N-1 tapes.

POLY tape
Strings are written to N-1 tapes, one at a time, until the lowest of the next level Fibonacci numbers is reached; then to N-2 tapes until the next number at the same level has been reached, and so on. When a complete level has been reached the next level numbers are calculated and the process is restarted. When all strings have been distributed dummy strings are added if necessary to make up Fibonacci numbers. The numbers of real and dummy strings are kept in PPITPTBL.

Module Table for Diagram N4.12

Function	Conditions	Mod
1a Sequence Distribution algorithm	CRCX	8ON
	OSCL	RON
	BALN (disk)	ROK
	BALN (tape)	ROI
	POLY	ROJ
1b Write	CRCX	8PA
	BALN (disk)	RPB
	Other	RPA

N3.2 Intermediate Merge (Phase 2)



Extended Description for Diagram N3.2

1. The selection of modules is made by ICERC7 using information from PPISW1 and from the module table TBLPH2RN*. Exit information is taken from the Exit Name Table (↑PPIAPGC). ICERC9 a) loads all running modules and user routines, then b) loads, executes and deletes the assignment modules sequentially.

2. Take exit E21*.
   Acquire main storage (through GETMAIN) for the phase work area and for buffers.
   Generate the record move routine.
   Build the input buffer table and network table in the phase work area*.
   Generate control blocks (DCB, IOB, ECB) and CCWs in the phase work area.
   Take exits E28 and E29*.
   Open the SORTWK data sets.
   Code in the running modules is initialized by the respective assignment modules.

Points 3, 4 and 5 are repeated until no input strings are left:

3. Read one string from each input tape--unless BALN (disk) is being used, in which case read M (or remaining, if less) strings at a time from one area (the largest of those with strings from the previous phase). For reading, the EXCP macro instruction is used.

4. Store the address of one record from each string (or its extracted control fields) in a network table (↑PPINETAR).
   Take exit E61.
   Find the next record to be moved to the output buffer, using binary search.
   Put the address of the next record in the 'winner's' string in the network table.
   Take exit E25.

5. For writing, the EXCP macro instruction is used.
   CRCX: The string is written to the area not used as input.
   OSCL: The string is written to the next tape.
   BALN (disk) : The string is written to the output area left empty by the previous phase or emptied in this pass.
   BALN (tape) : The string is written to the next tape.
   POLY: The string is written to the only available tape.

If the CRCX or OSCL technique is being used, the program returns from here to Phase 1 (see diagram N3.1), unless a) there is no more input, or b) the current level of strings now numbers M, in which case it repeats points 3-5.

6. Take exit E27.
   Check the record counts.
   Close the SORTWK data sets.
   ICERCT free main storage and deletes modules.

Unless the CRCX or OSCL technique is being used, the entire phase is re-executed until only one more merge pass is needed.

*When the CRCX or OSCL technique is used, these steps are executed in phase 1; see MO diagram N3.1

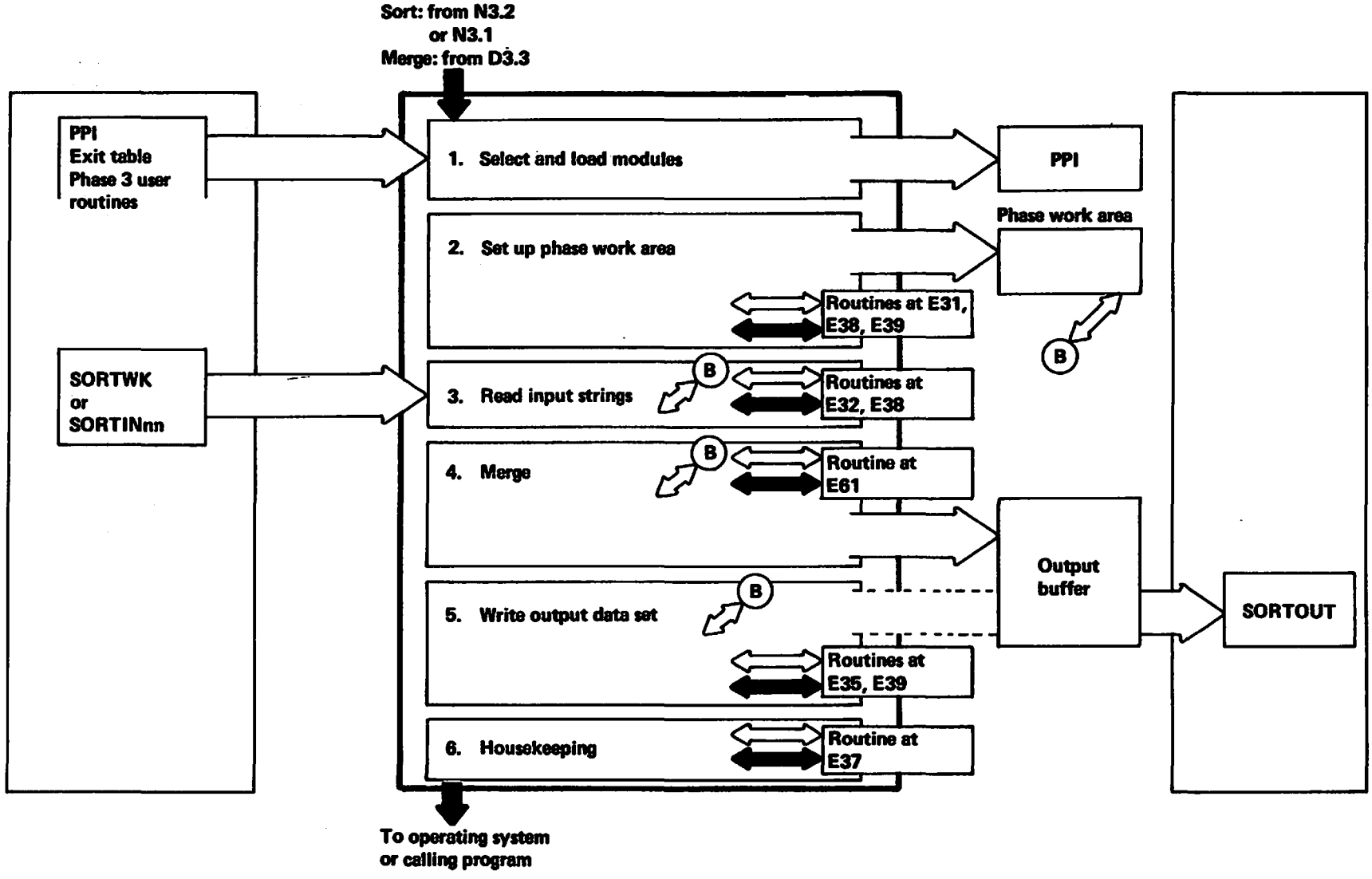
Module Table for diagram N3.2

Function	Conditions				Mod
1a Select modules	(Not CRCX or OSCL)				RC7
1b Load modules	Always				RC9
2a Allocate storage	(Not CRCX or OSCL)				APH
2b Build control blocks and tables	BALN (disk)				AGJ
	POLY,BALN (tape)				AGG
3 Read	POLY,BALN (t)			forward	RGL
				backward	RGB
	OSCL				RGB
	BALN (d)				RGC
	CRCX				8GB
4a Compare, select, update tree	Multiple				ROP
	Single				ROQ
4b Deblock/block	Fix	Exit	CRCX,OSCL		RBW
			Other		RBJ
		No E	256+	CRCX,OSCL	RBU
				Other	RBH
			<256	CRCX,OSCL	RBT
				Other	RBG
	Var	Exit	CRCX,OSCL		RBX
			Other		RBK
		No E	CRCX,OSCL		RBV
			Other		RBI
4c Move (additional to deblock/block for fix 256+ and/or with exits)					ABR[1]
5a Sequence distribution algorithm	CRCX				8ON
	OSCL				RON
	BALN (disk)				ROT
	BALN (tape)				ROR
	POLY				ROS
5b Write	CRCX				8PA
	OSCL				RPA
	BALN (tape) ,POLY				RPD
	BALN (disk)				RPE
6 Housekeeping	CRCX				8PM
	OSCL				RPM
	Other				RPF

[1]The move routine is generated by this module and kept in main storage (↑PPIBDSVA+8).

This page intentionally left blank

N3.3 Final Merge (Phase 3), or Merge Only



Extended Description for Diagram N3.3

1. The selection of modules is made by ICERC8 using information from PPISW1 combined with the module table TBLPH3RN.
   Exit information is taken from the Exit Name Table (.PPIAPGC).
   ICERC9 first loads the running modules and user routines, then loads, executes and deletes the assignment modules sequentially.

2. Take exit E31.
   Acquire main storage (through GETMAIN) for the phase work area and buffers.
   Generate the move routine in the phase work area.
   Build control blocks (DCB, IOB, ECB) and CCWs in the phase work area.
   Take exits E38 and E39.
   Open SORTOUT and the input data sets.
   Code in the running modules is initialized by the respective assignment modules.

3. For a sort, the strings are read from the SORTWK data sets.
   For a merge, the input strings are the SORTINnn data sets, or are supplied through E32.

4. Take exit E61.
   The address of one record from each input string (or of its extracted control fields) is kept in a network table (¢PPINETAR). The next record to be moved to the output buffer is determined using binary search.
   The address of the next record in the 'winner's' string is put in the network table.

5. Take exit E35. The record may be altered or deleted or a new record inserted. The current record is then written to SORTOUT. The program uses QSAM or VSAM for writing output.
   Normally locate mode is used, but move mode is used for spanned records.

6. Take exit E37.
   Check record counts.
   Close data sets, free main storage, and delete modules.

Module Table for Diagram N3.3

Function	Conditions				Mod
1a Select modules	Always				RC8
1b Load modules	Always				RC9
2a Allocate storage	Always				API
2b Build control blocks and tables	Tape sort				APK
	Disk sort				AGK
	Merge				APF
3a Read	POLY,BALN (tape)			forward	RGM
				backward	RGD
	OSCL				RGD
	BALN (disk)				RGE
	CRCX				8GC
	Merge-only			VSAM	RGV
				OSAM	- [1]
3b Deblock (sort)	Fix	POLY,BALN (t)		forwd	RDX
				back	RDH
		Other			RDH
	Var				RDI
Deblock (merge)	With E32				RDU
	No E32		spanned		8DJ
			other		RDJ
4 Merge	Single				ROQ
	Multiple				ROP
5a Block	Fix	Exit			RBM
		No E	256+		RBP
			<256		RBL
	Var	Exit	spanned		8BO
			other		RBO
		No E	spanned		8BN
			other		RBN
5b Write (additional to block for VSAM)					RPV
6 Housekeeping					RPG

[1] Carried out by the access method.

Section 3: Program Organization

This section describes the structure of the sort/merge program, and of each of the phases that make up the program. For a sorting application all phases are normally used, as shown in Figure 1 in Section 1. For merging applications, only the first and last phases are used.

The interface between the various load modules is shown.

The phases are described, and a cross reference given to the appropriate MO diagrams in the Method of Operation section (Section 2). Record movement is also described.

Phase structures and storage layouts are shown in figures at the end of this section.

Program Exit Handling

Provisions are contained in Phases 1, 2 and 3 for inclusion of user routines at program exits. The linkage editor includes the user routines in the load module for the appropriate phase; however, when proper specifications are provided on the MODS control statements, link-editing does not occur, and the routines are loaded by the phase loader modules. See Appendix A for the various exits.

Module Interface

The interface of load modules in the program is shown in Figure 7.

While ICERCB is in main storage a patch area of 48 bytes (hex 30) is available. It is located at offset 1132 (hex 46C) from the start of the PPI area. Register 13 points to the PPI. The area is initialized with binary zeros.

When Phase 1 of Blockset (ICEIPUT for FLR, ICEIPVT for VLR) detects an error condition, it hands control to either ICEMESI or ICEMSGV, respectively. Phase 3 hands control to ICEMESO. When error conditions are detected in Phase 1, 2, or 3 of the other techniques, control is handed to the phase interface module (ICEMON for Peerage and Vale techniques, and ICERCV for the other techniques). The phase interface module handles the issuing of the appropriate message.

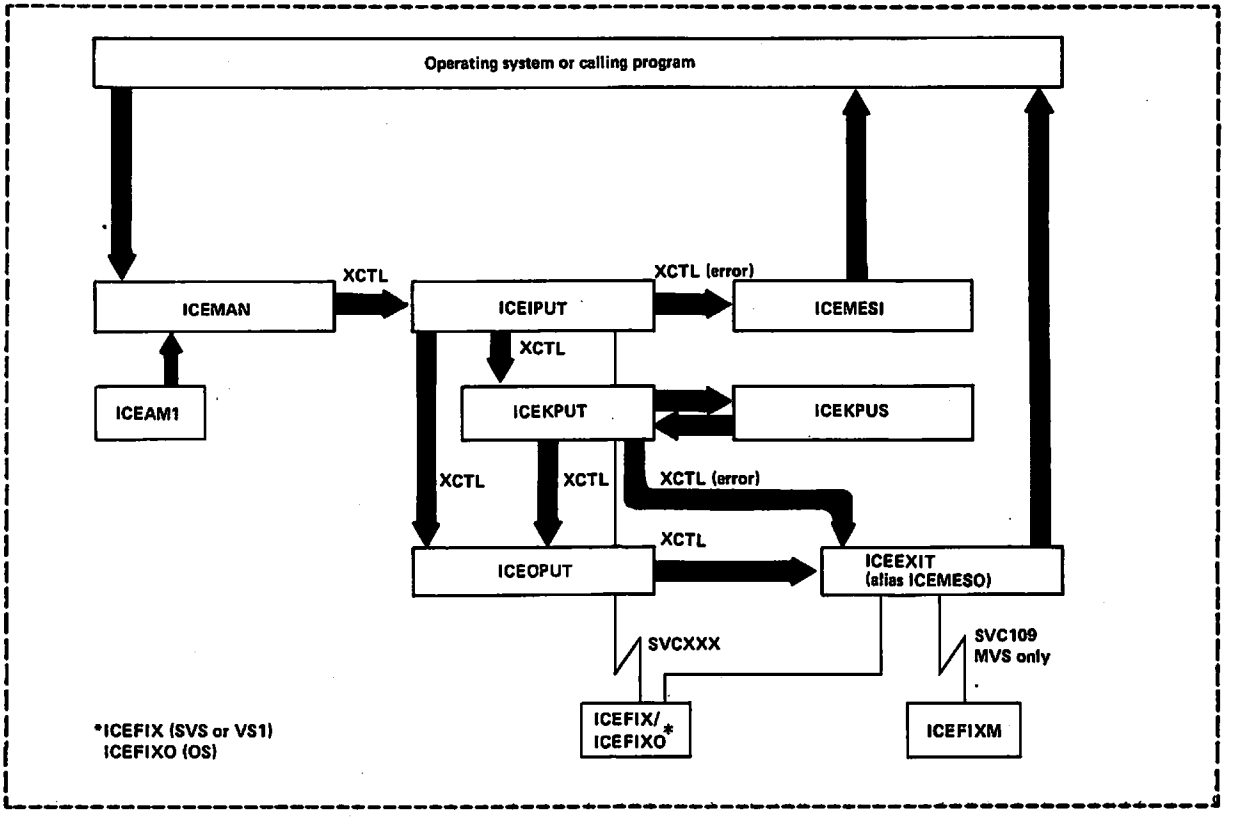


Figure 7. (1 of 4). Load Module Interface, FLR-Blockset

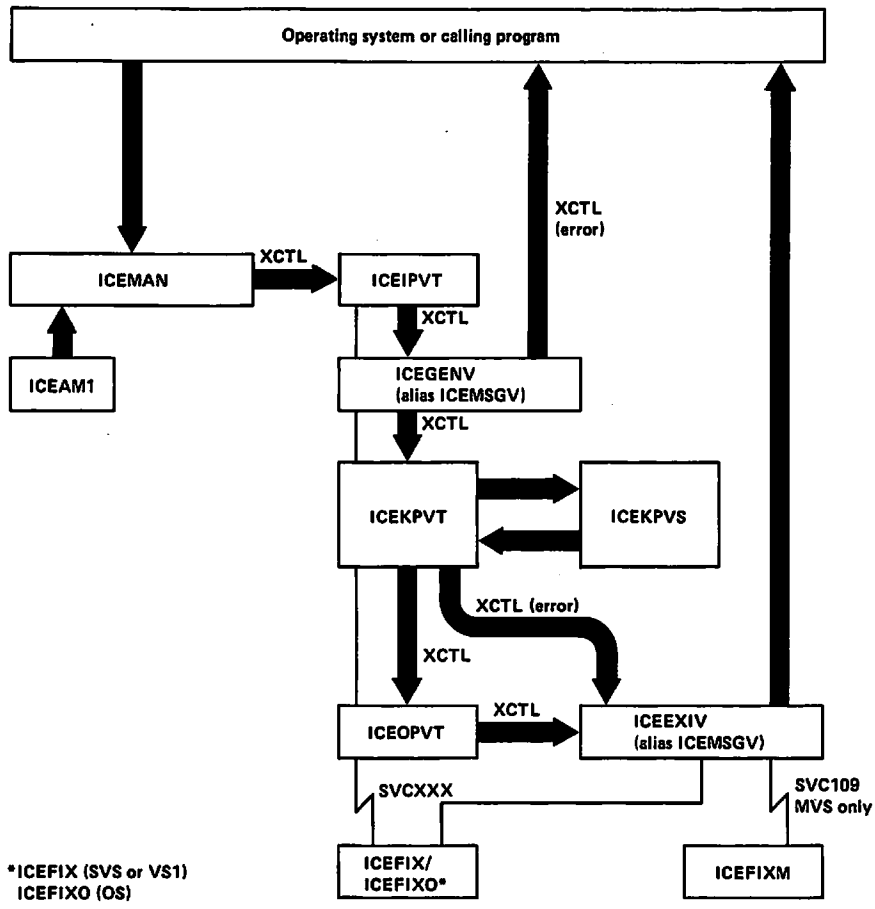


Figure 7. (2 of 4). Load Module Interface, VLR-Blockset

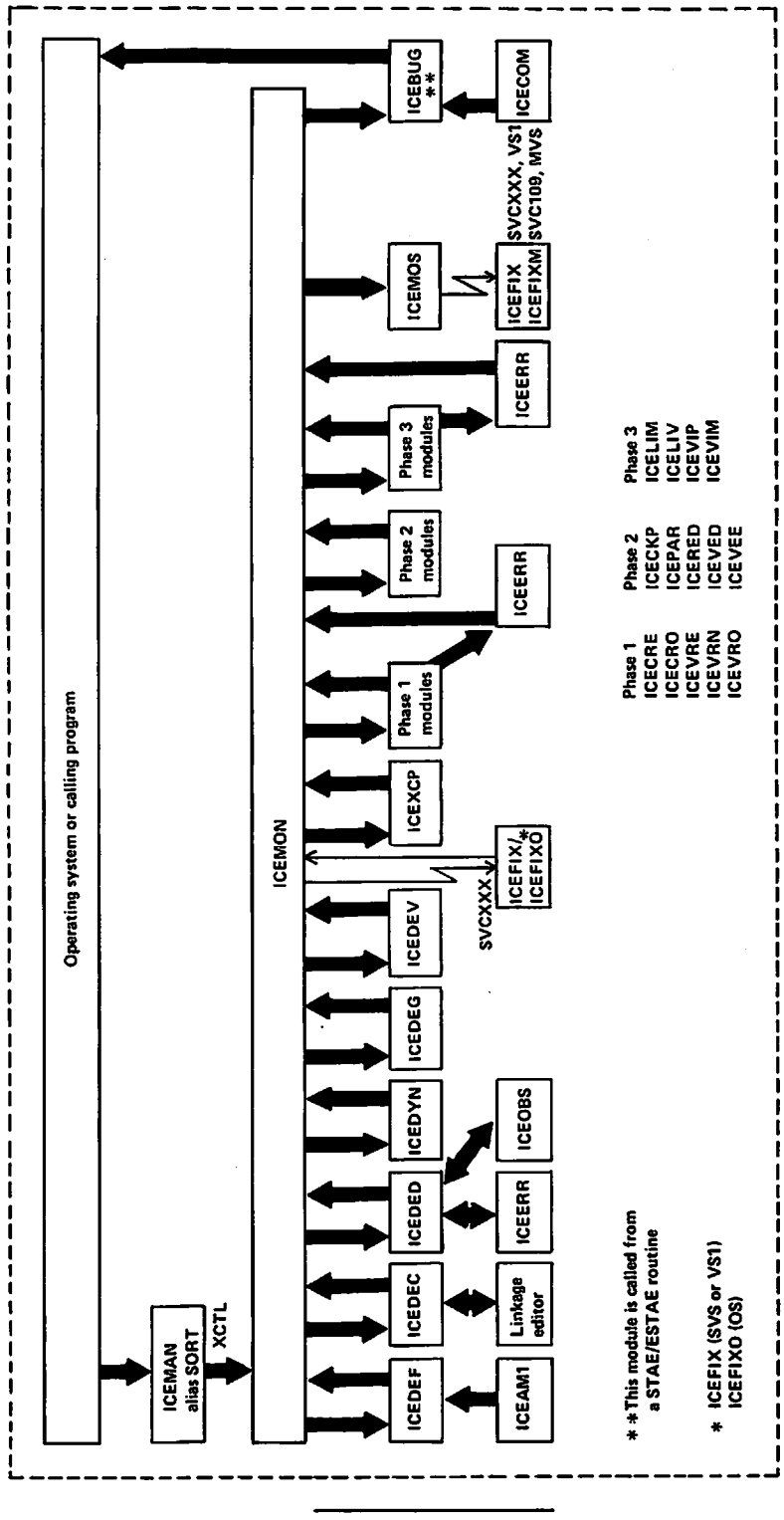


Figure 7. (3 of 4). Load Module Interface, Peerage and Vale

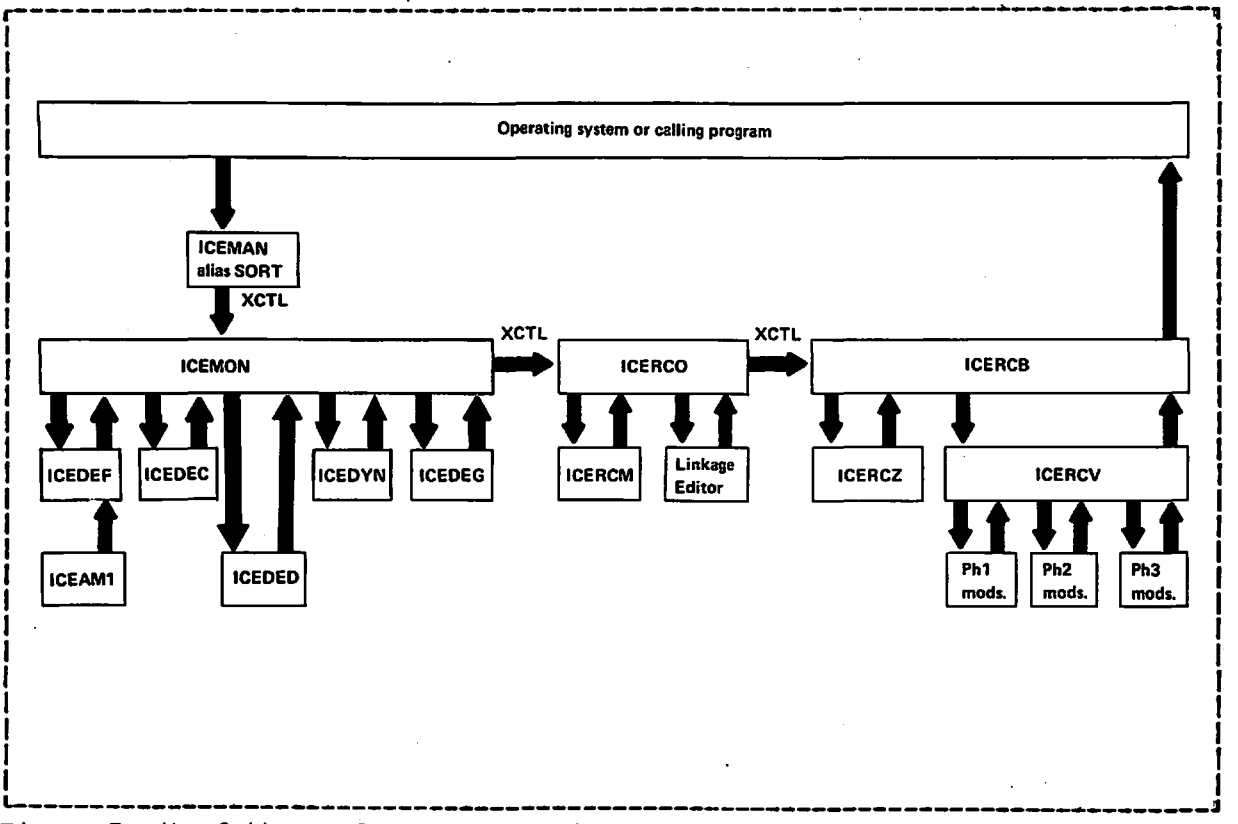


Figure 7. (4 of 4). Load Module Interface, Conventional Techniques

Phase 0

This phase collects the current system and sort control card information. It then decides which technique to be used and does the internal sort optimization. The functions are described in MO diagram D2. For conventional techniques, an SMF record type 16 is written to the system SMF data set, if requested.

Phase 1

Phase 1 performs the initial ordering of the input records. This is a one-pass phase (that is, each record is processed only once) that arranges the records of the input data set into ordered sequences. The sequences are written on the intermediate storage devices according to a predetermined distribution procedure.

With Blockset and Peerage, but not with Vale, an index entry is created for each work file block. This entry contains the control word of the first record in the block, and the block's disk address.

For Blockset applications this phase is described in MO diagram B3.1; for Peerage and Vale in MO diagram P3.1; for other techniques in MO diagram N3.1.

The input of records is controlled by EXCP, or by the SAM access method, and output is controlled by the sort/merge program--unless the application is a standard disk sort which needs no work storage, in which case EXCP or SAM is used to produce the output data set. The sorting method used to order the records into sequences is a version of the replacement-selection technique.

When the conventional OSCL tape or CRCX disk techniques are used, control alternates between Phase 1 and Phase 2. With the BALN tape or disk technique, Phase 1 is first completed and then control is handed to either Phase 2 or Phase 3.

Phase 2

BLOCKSET TECHNIQUE

This phase processes the index entries from Phase 1 and assigns each block to a cylinder block set for merging. Two construction algorithms are used: one for a merge which writes only to the output file, and one for a merge which also writes to the work file.

This phase is described in MO diagram B3.2.

OTHER TECHNIQUES

For all other techniques, this phase combines the sequences produced by Phase 1 into a lesser number of longer sequences. It makes as many passes as necessary until the number of record sequences resulting from a given pass is equal to or less than the maximum Phase 3 merge order.

The merge order (M) is the number of sequences merged in any one merge pass. Maximum M for a given application is limited by the number of intermediate storage areas available, by the technique being used, and frequently by the number of buffers which can be accommodated in available main storage.

Phase 2 continues, therefore, until the total number of sequences remaining on all work areas is less than or equal to maximum M for the application in question.

Peerage Technique

This phase is described in MO diagrams P3.2 and P3.3.

The phase is in two parts, carried out by modules ICEPAR and ICERED. ICEPAR receives control after Phase 1 (ICECRE). It partitions the files, i.e. rearranges their indexes so that they will be read in by the merge in the most efficient order. If necessary, ICEPAR then initiates an intermediate merge: it instructs ICEMON to hand control to ICERED, which after merging returns to ICEPAR via ICEMON. This process continues until ICEPAR detects that the program is ready to enter Phase 3.

Vale Technique

This phase is described in MO diagram P3.5.

Each pass can merge up to 64 sequences. Two output buffers are used. The number of input buffers is M (merge order) + 1; the extra one is for 'smart lookahead', which means that when a buffer from a given sequence is emptied its next block should already be in main storage.

Conventional Techniques

This phase is described in MO diagram N3.2.

Each pass is capable of merging up to 16 previously sorted record sequences, or 63 if the balanced disk technique is used.

When the OSCL or CRCX techniques are used, control alternates between Phase 1 and Phase 2.

Phase 3

Phase 3 is a one-pass phase and produces a single ordered record sequence, thus completing the application.

For Peerage this phase is described in MO diagram P3.4; for Vale in P3.6; for Blockset in MO diagram B3.3; and for other techniques in N3.3.

In a sorting application, it follows Phase 2 (or Phase 1 if a Vale, Blockset, or conventional sort, skips Phase 2). In a merging application, it is executed immediately after Phase 0.

For a merging application, record input and output operations are performed by the control program's data management routines. For the final merge of a sorting application, Sort handles the record input operation, and EXCP or SAM is used to handle the record output operation, as in a merging case.

At the end of this phase for Blockset, Peerage, and Vale, an SMF record type 16 is generated and written to the system's SMF data set, if requested as an installation option.

Phase Structures

The Blockset, Peerage, and Vale technique phases consist normally of one load module each.

With other techniques the load modules are either planned overlay structures or dynamic structures. The organization of each phase with planned overlay or dynamic structures, and the modules that comprise them, are presented on the following pages. The modules for each phase are not necessarily placed in contiguous storage locations. In the case of dynamic phase structures, the modules are presented in the order in which they receive control.

REENTERABLE LOAD MODULES, Peerage in SYS1.LPALIB, SYS1.LINKLIB, or a private link library	
ICEAM1	ICEERR
ICECKP	ICELIM
ICECRE	ICELIV
ICECRO	ICEMAN* alias sort
ICEDEC	ICEMON
ICEDED	ICEMOS
ICEDEF	ICEOBS
ICEDEG	ICEPAR
ICEDEV	ICERED
ICEDYN	ICEXCP
NONREENTERABLE LINK MODULES, Peerage in SYS1.LINKLIB or a private link library.	
ICEBUG (used for producing a special formatted dump). ICECOM (source code for ICECOMMA.	
SVC MODULES, REENTERABLE, Peerage in SYS1.LPALIB or SYS1.SVCLIB.	
IGX00017: ICEFIX for SVS/VS1 ICEFIXM for MVS SVCXXX: ICEFIXO for MFT/MVT	
The SVC names can be changed by the user for SVS/VS1, and must be renamed by the user for MFT/MVT.	
*See Blockset modules for further information.	

Figure 8. Peerage Modules

REENTERABLE LOAD MODULES, Vale in SYS1.LPALIB, SYS1.LINKLIB, or a private link library		
ICEAM1	ICEMON	ICEVRO
ICECKP	ICEMOS	ICEXCP
ICEDEC	ICEOBS	
ICEDED	ICEVED	
ICEDEF	ICEVEE	
ICEDEG	ICEVIM	
ICEDEV	ICEVIP	
ICEDYN	ICEVRE	
ICEERR	ICEVRN	
ICEMAN* alias sort		
NONREENTERABLE LINK MODULES, Vale in SYS1.LINKLIB or a private link library.		
ICEBUG (used for producing a special formatted dump). ICECOM (source code for ICECOMMA.		
SVC MODULES, REENTERABLE, Vale in SYS1.LPALIB or SYS1.SVCLIB.		
IGX00017: ICEFIX for SVS/VS1 ICEFIXM for MVS SVCXXX: ICEFIXO for MFT/MVT		
The SVC names can be changed by the user for SVS/VS1, and must be renamed by the user for MFT/MVT.		
*See Blockset modules for further information.		

Figure 9. Vale Modules

OJECT AND LOAD MODULES, FLR-Blockset in SYS1.LPALIB, SYS1.LINKLIB, or a private link library	
Load Modules	Object Modules
ICEMAN alis SORT	ICEMAN ICEINIB ICEINIC ICEINID ICEINI0 ICEDYNA ICEMESS ICEEXIN ICEFAUL ICEDEVT ICEMSGN ICEFORM ICEFVLN
ICEAM1	ICEAM1
ICEIPUT	ICEXPUT ICEE15B ICEFILI ICEIPUT ICEIPUM ICEIPUB ICECYLN ICEDEVT ICEEXII
ICEMESI	ICEMESI ICEMESS ICEEXII ICEMSGI ICESUBS
ICEKPUT	ICEKPUT ICEKPUV ICEKPUA ICEKPUB ICEMESS ICEEXIK ICEFILK ICECYLN ICEDEVT ICEMSGK ICEXPUT
ICEKPUS	ICEKPUS
SVC MODULES, REENTERABLE, FLR-Blockset in SYS1.LPALIB or SYS1.SVCLIB	
IGX00017: ICEFIX for SVS/VS1 ICEFIXM for MVS SVCXXX: ICEFIXO for MFT/MVT	
The SVC names can be changed by the user for SVS/VS1, and must be renamed by the user for MFT/MVT	

Figure 10. (1 of 2). FLR-Blockset Modules

OJECT AND LOAD MODULES, FLR-Blockset in SYS1.LPALIB, SYS1.LINKLIB, or a private link library	
Load Modules	Object Modules
ICEOPUT	ICECOBU ICEE35B ICEFILO ICEOPUT ICEOPUA ICEOPUB ICELIMI ICECYLN ICEDEVT
ICEEXIT, alias ICEMES0	ICECKUT ICEDEVT ICEEXIO ICEEXIT ICEMESS ICEMSGO ICESUBS ICEXPUT
SVC MODULES, REENTERABLE, FLR-Blockset in SYS1.LPALIB or SYS1.SVCLIB	
IGX00017: ICEFIX for SVS/VS1 ICEFIXM for MVS SVCXXX: ICEFIXO for MFT/MVT	
The SVC names can be changed by the user for SVS/VS1, and must be renamed by the user for MFT/MVT	

Figure 10. (2 of 2). FLR-Blockset Modules

OJECT AND LOAD MODULES, VLR-Blockset in SYS1.LPALIB, SYS1.LINKLIB, or a private link library	
Load Modules	Object Modules
ICEMAN alis SORT	Same as FLR-BLOCKSET
ICEAM1	ICEAM1
ICEIPVT	ICEXPUT ICECOBV ICE15V ICEFVLI ICEIPVT ICEIPVM ICEIPVB ICECYLN ICEDEVT
ICEGENV alias ICEMESV	ICEGENV ICEMESV ICEXII ICEXPUT ICESUBX
ICEKPVT	ICEKPVT ICEKPVV ICEKPVA ICEKPVB ICEMESS ICEEXIK ICECYLN ICEMSGK ICEXPUT ICEDEVT ICEFVLK
ICEKPVS	ICEKPVS
ICEOPVT	ICE35VL ICEFVLO ICEOPVT ICEOPVA ICEOPVB ICECYLN ICEDEVT
ICEEXIV alias ICEMSGV	ICEOXIV ICEOXOV ICEMESS ICEMSGO ICEXPUT ICESUBX ICEEXIV
SVC MODULES, REENTERABLE, VLR-Blockset in SYS1.LPALIB or SYS1.SVCLIB	
IGX00017: ICEFIX for SVS/VS1 ICEFIXM for MVS SVCXXX: ICEFIXD for MFT/MVT	
The SVC names can be changed by the user for SVS/VS1, and must be renamed by the user for MFT/MVT	

Figure 10A. VLR-Blockset Modules

NONREENTERABLE MODULES, Conventional Techniques in SYS1.LINKLIB or a private link library

ICERCB	ICERCM (overlay module, see Figure 12)
ICERCO	ICERCZ (overlay module, see Figure 12)

'SORT' MODULES, NONREENTERABLE, Conventional Techniques in SYS1.SORTLIB or a private sort library

ICEABA	ICEADL	ICEAP2	ICEACQ	ICERDH	ICEFCN	ICE9GN
ICEABB	ICEADP	ICEAP3	ICEACR	ICERDI	ICERCP	ICE9ON
ICEABC	ICEADQ	ICECHK	ICEACS	ICERDJ	ICERCQ	ICE9PA
ICEABE	ICEADR	ICEDM4	ICEACT	ICERDL	ICERCR	
ICEABF	ICEADS	ICEEX1	ICEACW	ICERDP	ICERCS	
ICEABG	ICEADT	ICEEX2	ICEACX	ICERDQ	ICERCT	
ICEABH	ICEADU	ICEEX3	ICERBH	ICERDR	ICERCW	
ICEABI	ICEADX	ICEFBA	ICERBI	ICERDS	ICERCX	
ICEABJ	ICEAGA	ICERBB	ICERBJ	ICERDT	ICERCY	
ICEABK	ICEAGB	ICERBC	ICERBK	ICERDU	ICERCZ	
ICEABL	ICEAGC	ICERBE	ICERBL	ICERDX	ICERPA	
ICEABM	ICEAGD	ICERBF	ICERBM	ICERGB	ICERPB	
ICEABN	ICEAGE	ICERBG	ICERBN	ICERGC	ICERPC	
ICEABO	ICEAGF	ICEAGK	ICERBO	ICERGD	ICERPD	
ICEABP	ICEAGG	ICEAGL	ICERBP	ICERGE	ICERPE	
ICEABQ	ICEAGH	ICEAGM	ICERBT	ICERGF	ICERPF	
ICEABR	ICEAGI	ICEAGN	ICERBU	ICERGL	ICERPG	
ICEABS	ICEAGJ	ICEAMA	ICERBV	ICERGM	ICERPM	
ICEABT	ICEAOY	ICEAMB	ICERBW	ICERGV	ICERPV	
ICEABU	ICEAOZ	ICEAMC	ICERBX	ICERMA	ICE8BN	
ICEABV	ICEAPA	ICEAOA	ICERBY	ICERMB	ICE8BO	
ICEABW	ICEAPB	ICEAOB	ICERBZ	ICERMC	ICE8CR	
ICEABX	ICEAPC	ICEAOC	ICERCT	ICERQA	ICF8DJ	
ICEABY	ICEAPD	ICEAOD	ICERCV	ICERQB	ICE8GB	
ICEABZ	ICEAPE	ICEAOE	ICERC6	ICERCC	ICE8GC	
ICEADB	ICEAPF	ICEAOF	ICERC7	ICERCD	ICE8ON	
ICEADC	ICEAPG	ICEAOG	ICERC8	ICERCE	ICE8PA	
ICEADD	ICEAPH	ICEAOH	ICERC9	ICERQF	ICF8PM	
ICEADE	ICEAPI	ICEAOI	ICERDB	ICERCG	ICE9BN	
ICEADG	ICEAPJ	ICEAOJ	ICERDC	ICERCH	ICE9BO	
ICEADH	ICEAPK	ICEAOK	ICERDD	ICERCI	ICE9DJ	
ICEADI	ICEAPL	ICEAON	ICERDE	ICERCJ	ICE9GB	
ICEADJ	ICEAP1	ICEAOP	ICERDG	ICERCK	ICE9GC	

Figure 11. Conventional Technique Modules

Load Module	Object Modules		
ICERCM	ICERCM	ICERCH	ICERCN
	ICERCU	ICERCX	ICEBGB
	ICERCQ	ICERCP	ICERCS
	ICERCD	ICERCI	ICERCL
	ICERC3	ICERC2	
ICERCZ	ICERCZ	ICEAO4	ICERCJ
	ICERCU	ICEAO5	ICEAO1
	ICERC1	ICERC4	ICEAO2
	ICEAOL	ICERCK	
	ICEAOM	ICE8CK	

Figure 12. Object Modules in the Two Overlay Load Modules

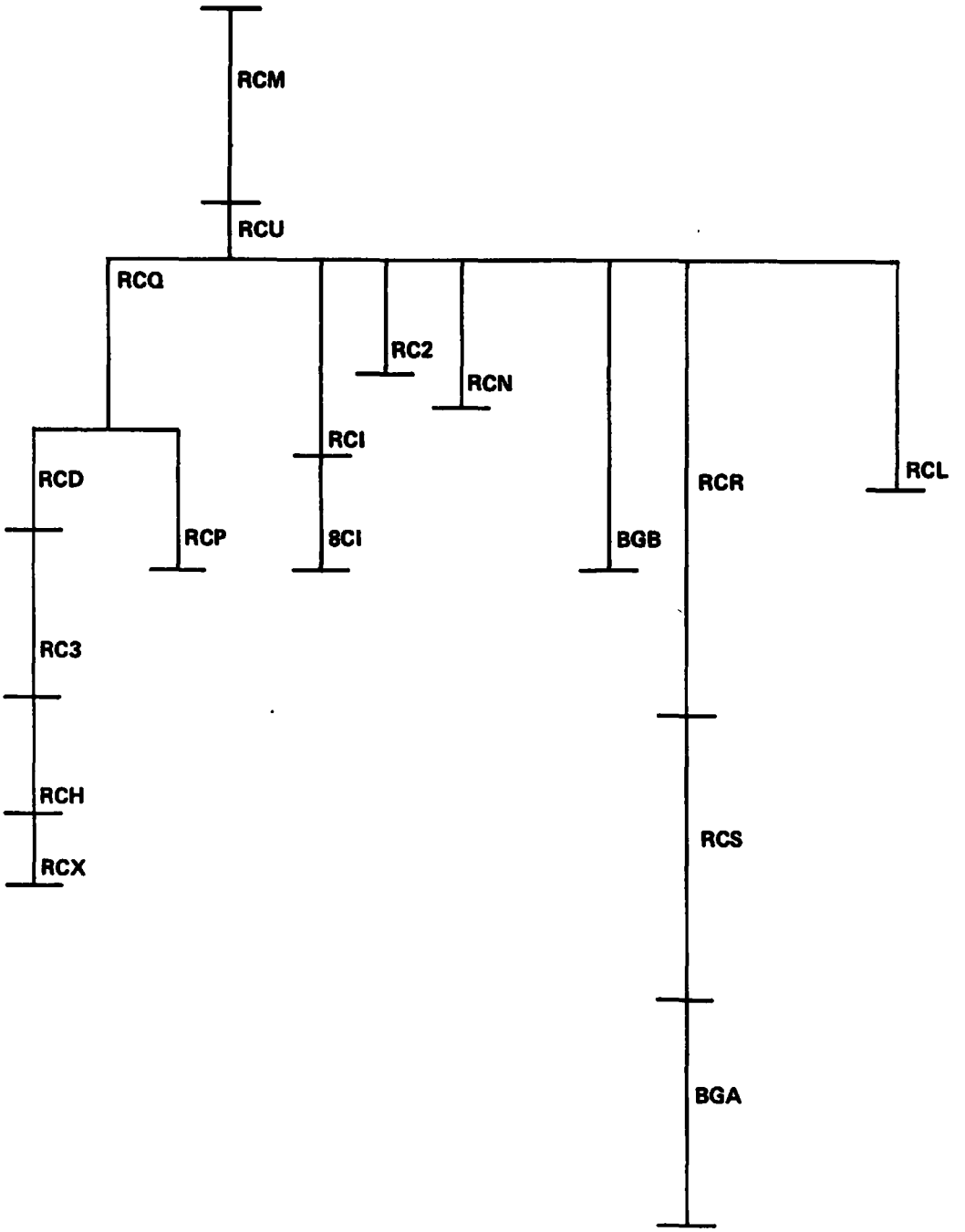


Figure 13. (1 of 6) Load Modules With Conventional Techniques
Phase 0, Initialization, Stage 2 (Definition)
Overlay Structure

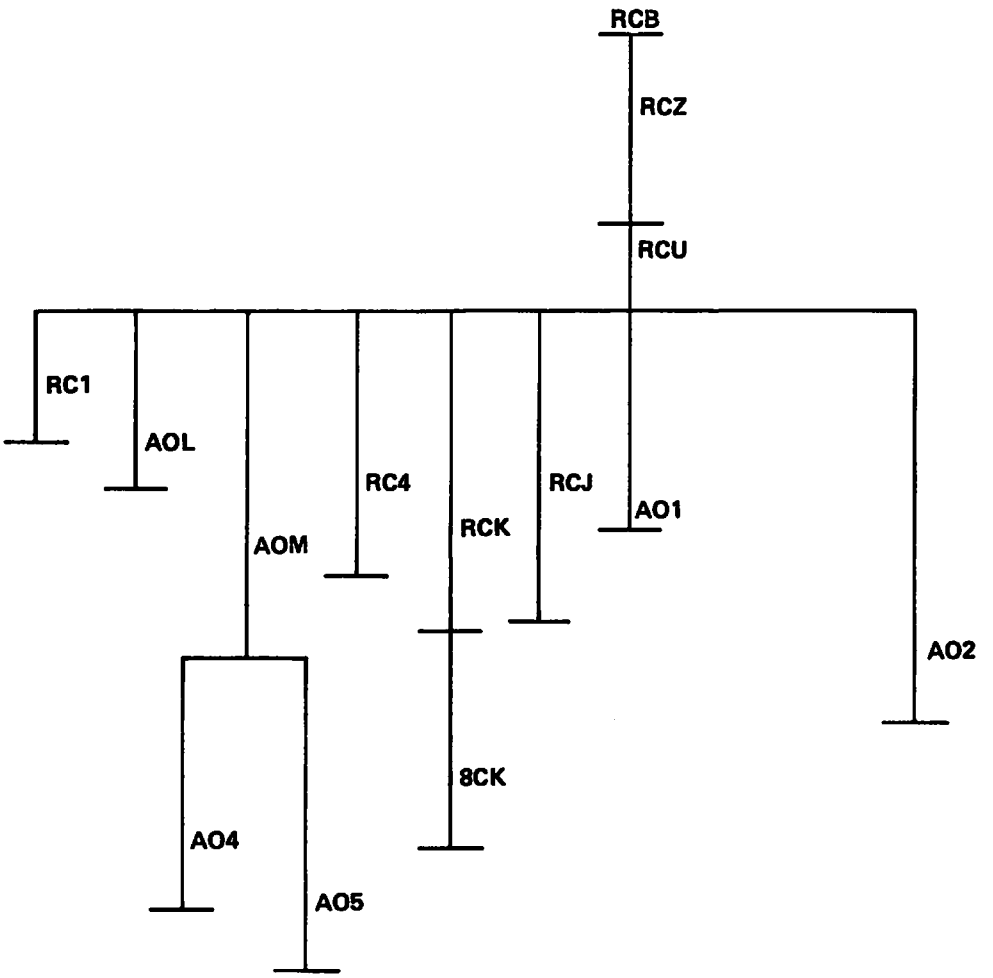


Figure 13. (2 of 6) Load Modules With Conventional Techniques
Phase 0, Initialization, Stage 2 (Optimization)
Overlay Structure

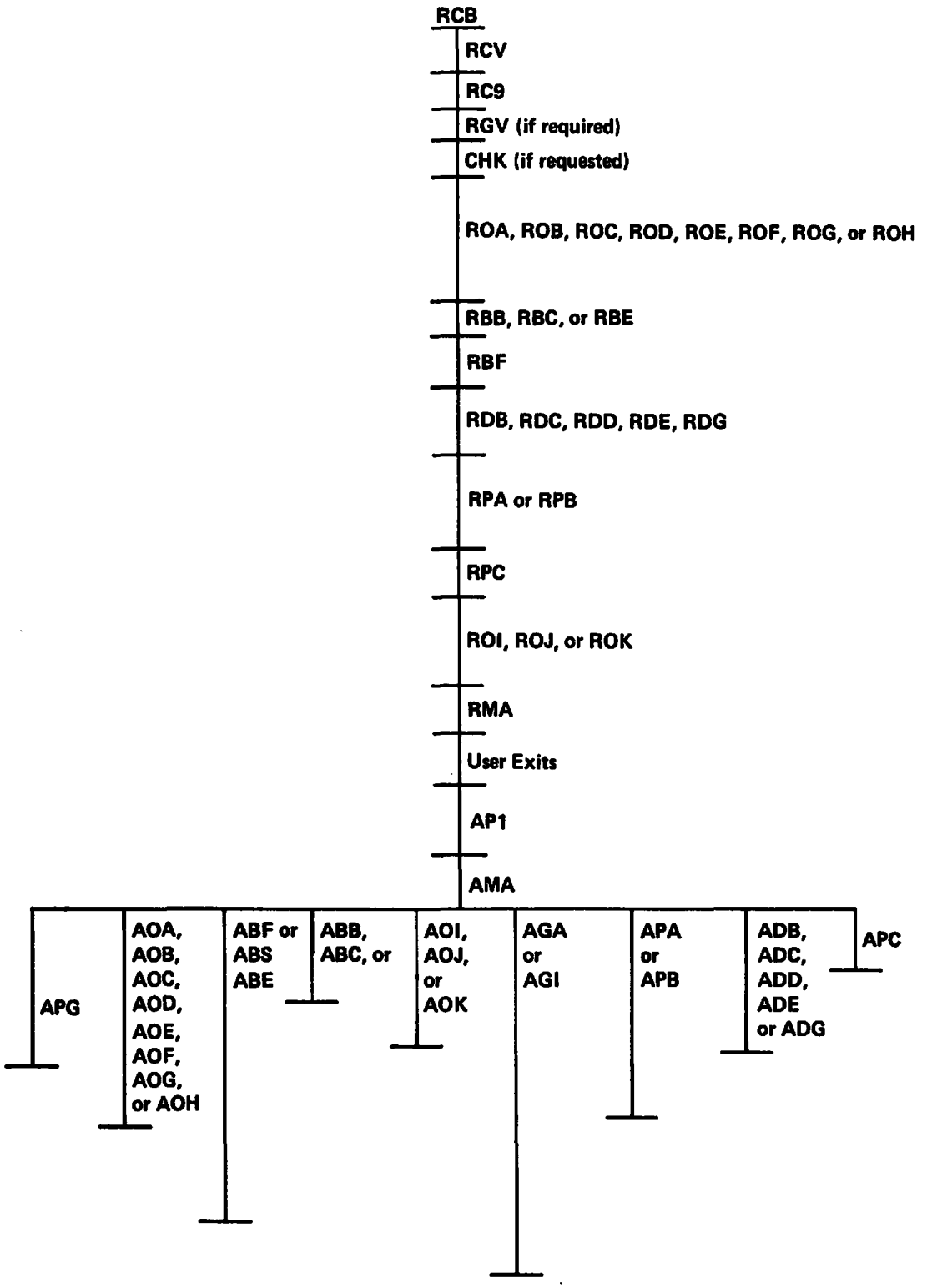


Figure 13. (3 of 6) Load Modules with Conventional Techniques Phase 1, BALN and POLY Techniques. Overlay Structure.

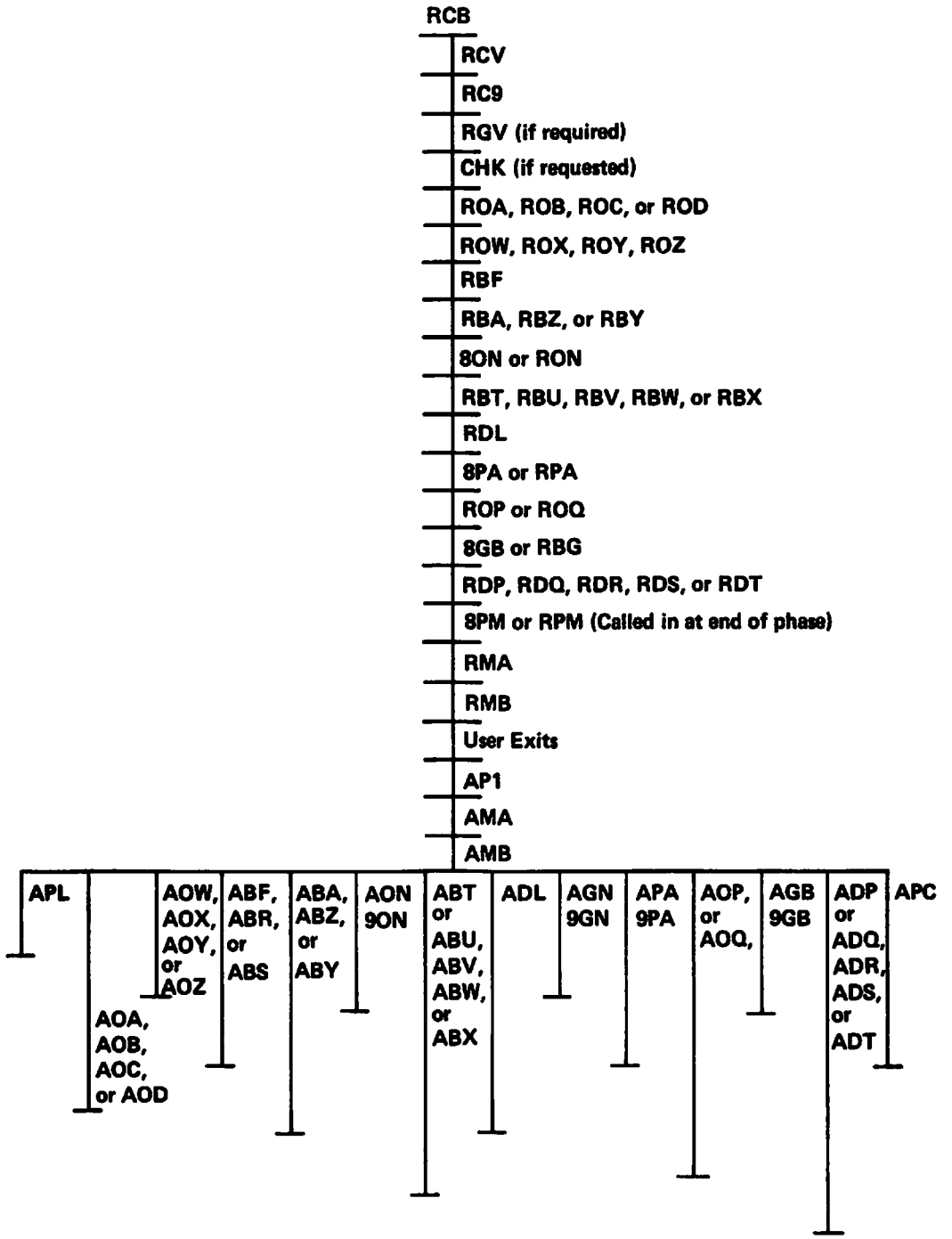


Figure 13. (4 of 6) Load Modules with Conventional Techniques Phases 1 and 2, OSCL and CRCX Techniques. Dynamic.

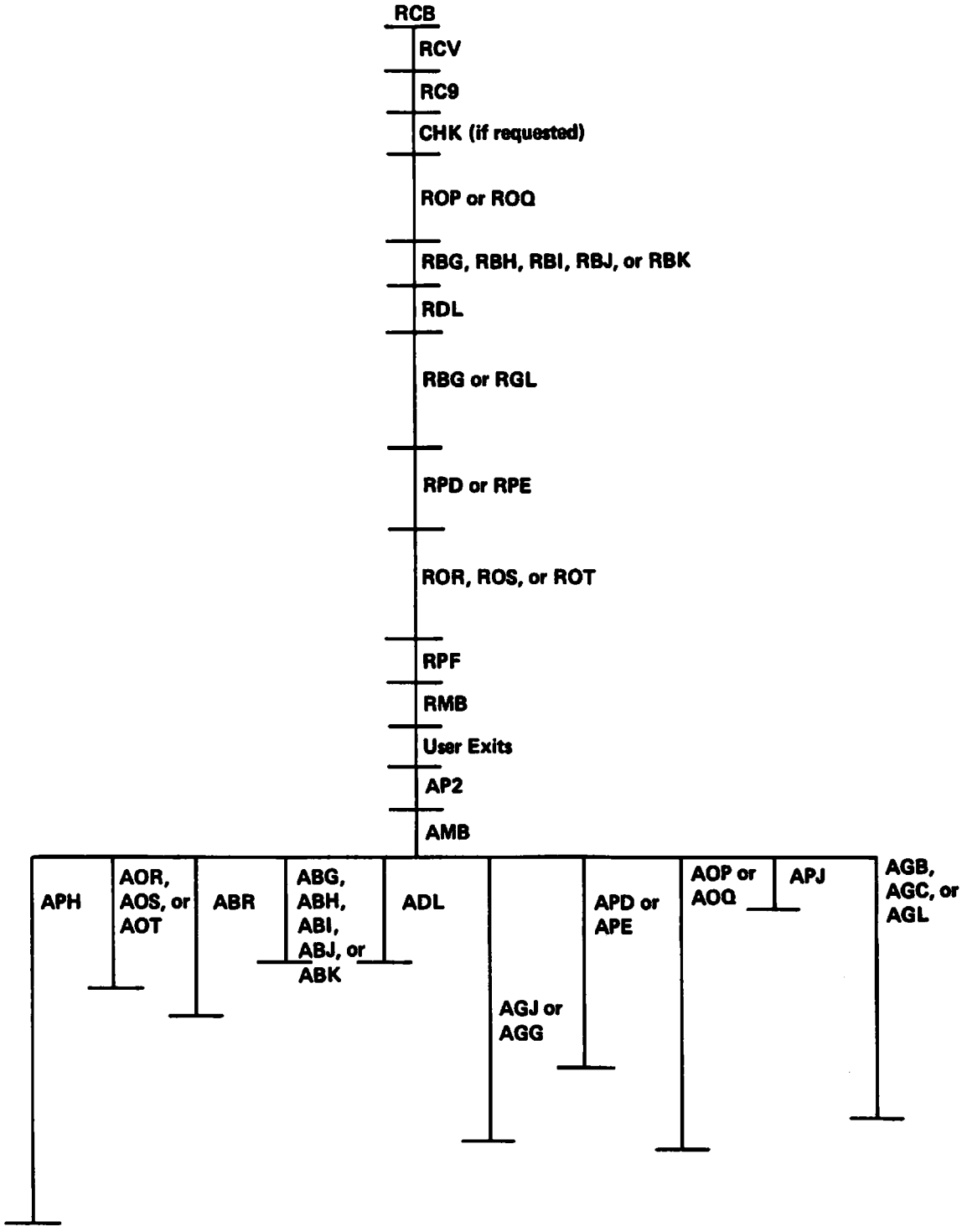


Figure 13. (5 of 6) Load Modules with Conventional Techniques Phase 2, BALN and POLY Techniques. Dynamic.

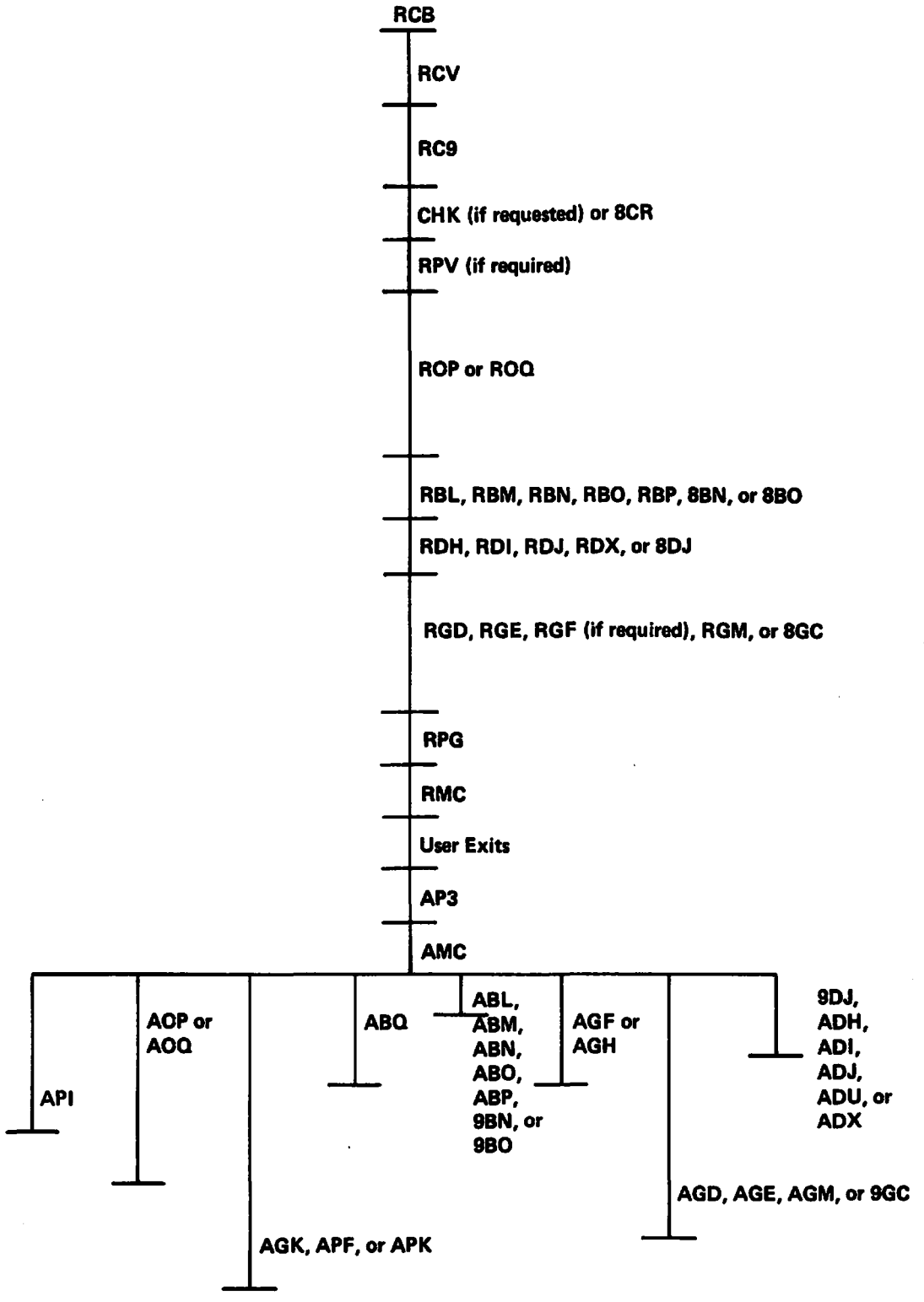


Figure 13. (6 of 6) Load Modules with Conventional Techniques
Phase 3.
Dynamic.

Storage Layouts

BLOCKSET, PEERAGE, AND VALE TECHNIQUES

Storage is allocated as follows:

1. Phase 0 issues a GETMAIN primarily for the communications area, COMMON or COMMA, and the extract/restore code area, which is variable in length.

2. If SORTWK data sets have been defined, a GETMAIN is issued for the necessary I/O area. Storage is also allocated to the Checkpoint/Restart track pool if necessary (though not for Blockset).

3. End of Phase 0 frees whatever part of the first area is not needed by COMMON or COMMA and the extract/restore code area.

4. Phases 1 and 3 begin by getting space for buffers for SORTIN and SORTOUT.

5. Each of the Phases 1-3 gets the remaining required storage by issuing a series of variable GETMAINS until it reaches SIZE or MAXLIM. At the end of the phase it frees this storage again.

OTHER TECHNIQUES

The storage allocation technique used is such that main storage is first allocated for necessary or required areas (proceeding from the largest to the smallest), and then for the optional areas (in priority order). This is accomplished by issuing variable-type requests for main storage. The minimum request is equal to the size of the largest necessary area; and the maximum, equal to the total size of all areas.

If additional main storage is needed, the unused portion of the previous allocation is made available for subsequent allocations. The minimum request becomes equal to the size of the area next-in-line to be allocated; and the maximum, equal to the total size of all unallocated areas. (This process is continued until either all areas are allocated, or the minimum request cannot be satisfied.)

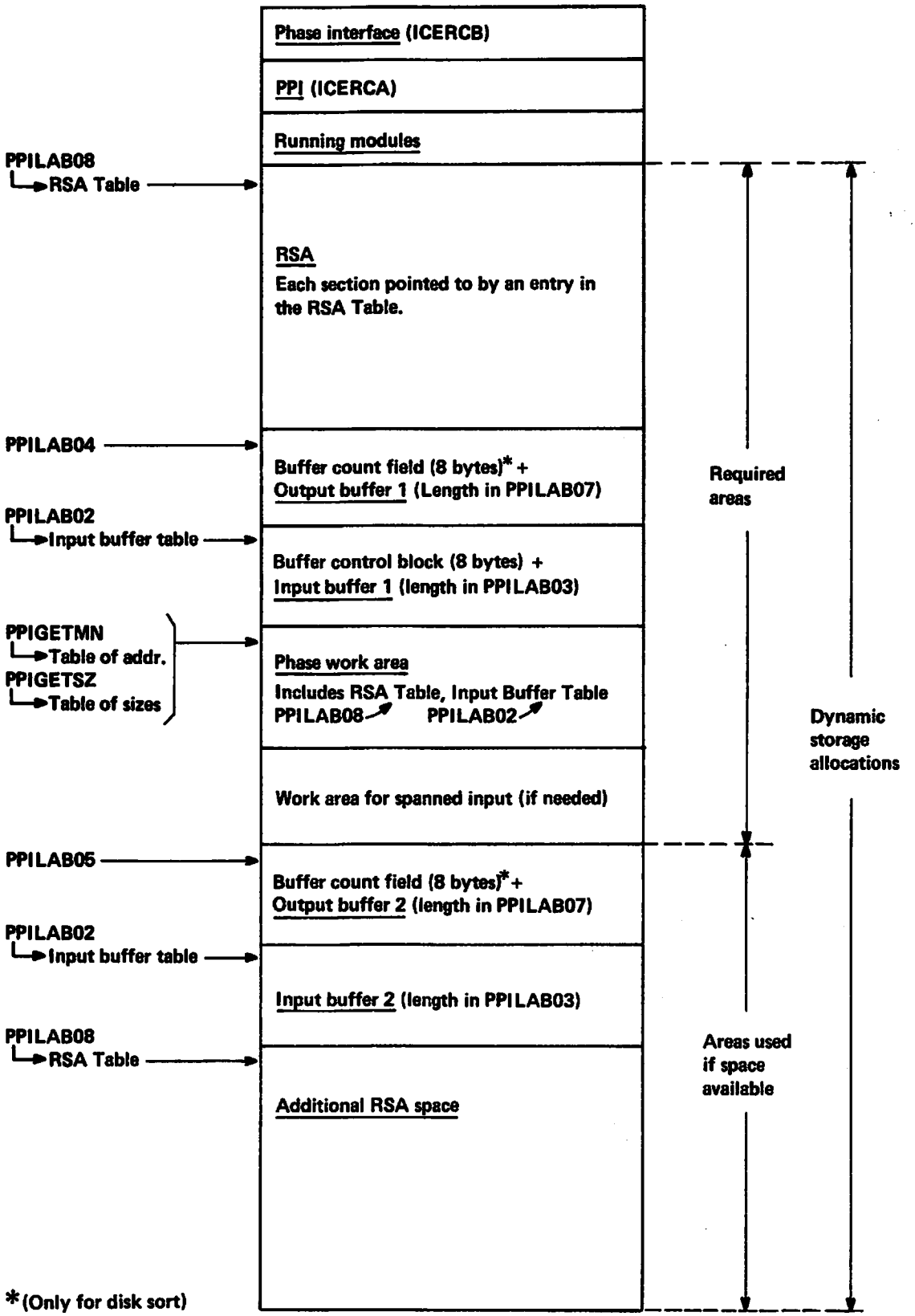


Figure 14. (1 of 4) Storage Layouts for Conventional Techniques Phase 1, BALN and POLY Techniques

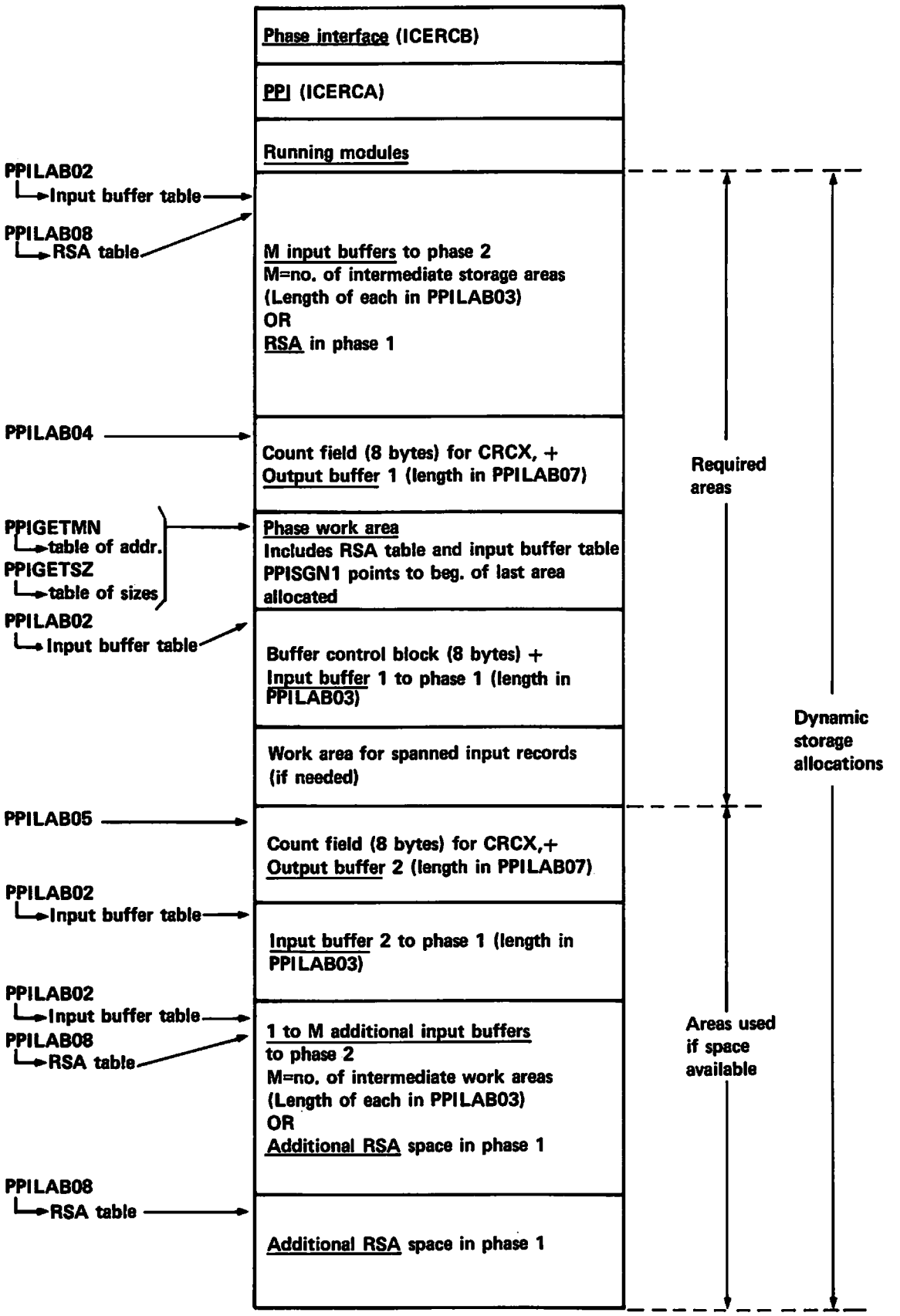


Figure 14. (2 of 4) Storage Layouts for Conventional Techniques Phases 1 and 2, CRCX and OSCL Techniques

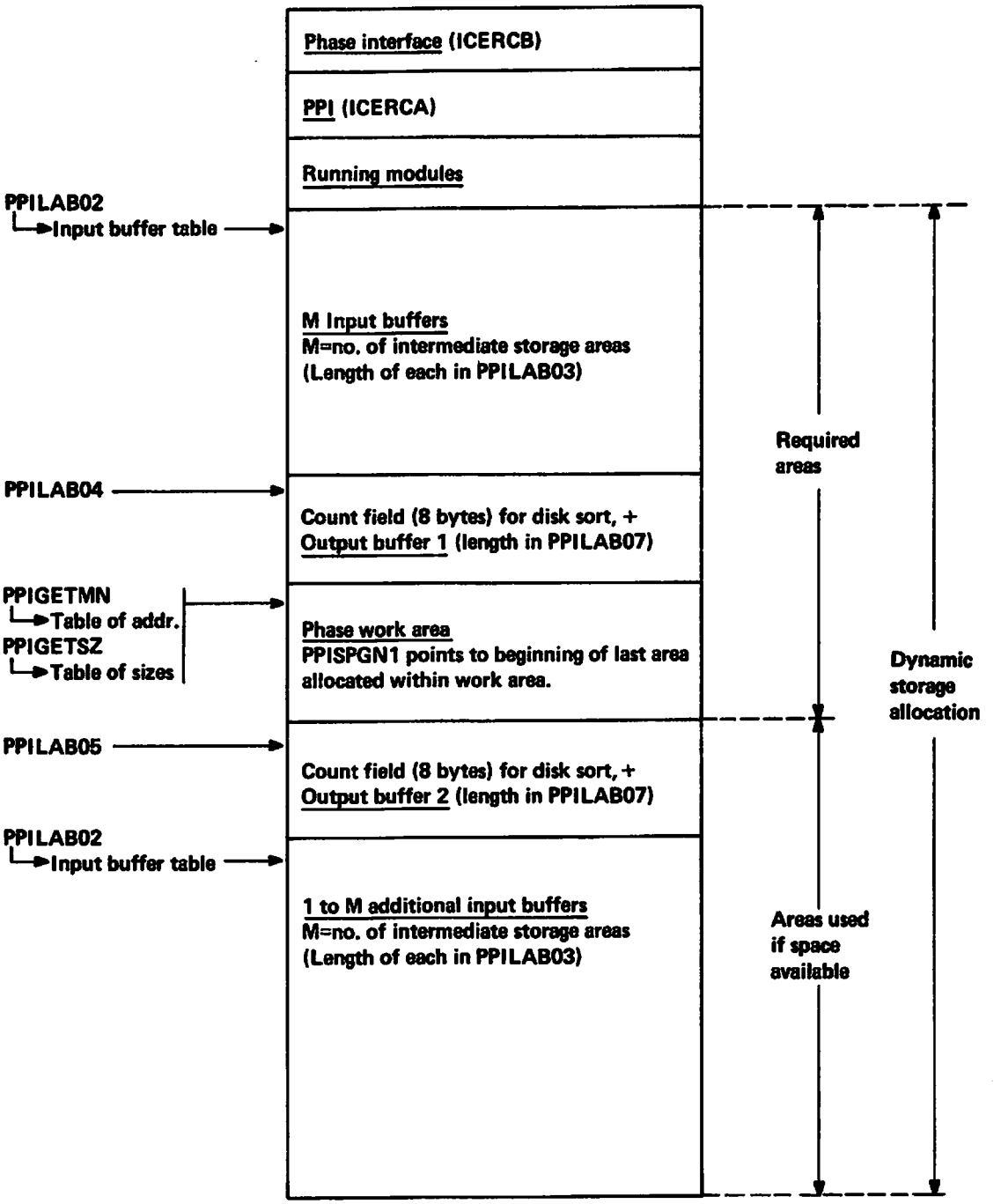


Figure 14. (3 of 4) Storage Layouts for Conventional Techniques Phase 2, POLY and BALN Techniques

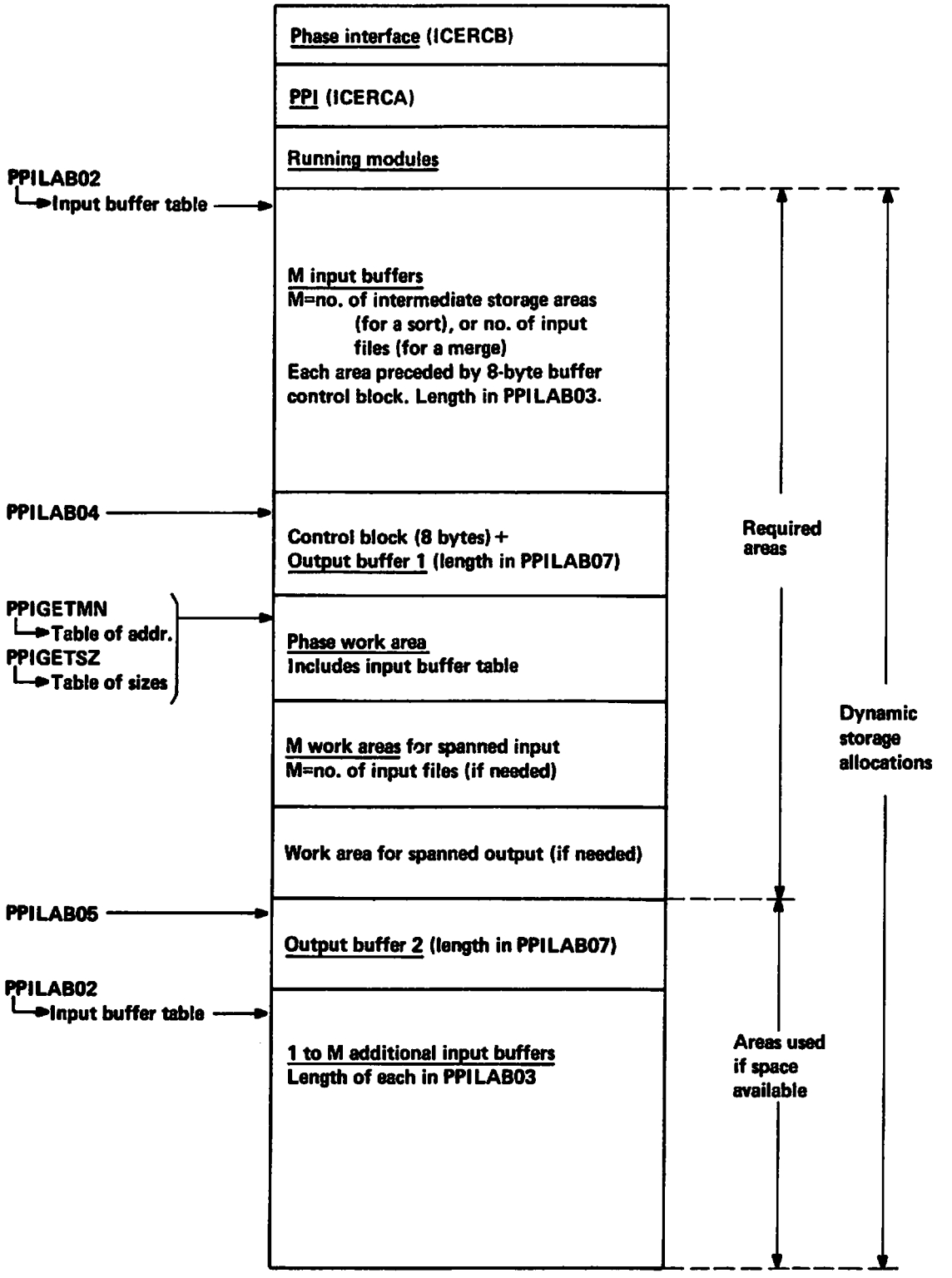


Figure 14. (4 of 4) Storage Layouts for Conventional Techniques Phase 3, All Techniques

Section 4. Directories

This section lists all module names: first Blockset, then Peerage and Vale, then conventional technique modules. Each list is in alphabetical order. Cross references are given to other parts of the publication.

Blockset Directory

This directory lists Blockset object modules. Their names are in the form ICEmod, where mod is the three or four character identification of the module. Only these last characters are listed. The list also gives a reference to the Method of Operation diagrams (MO diag) provided in Section 2.

Module Name	Used in Phase:	FLR or VLR	MO Diag	Purpose
AM1	0	FLR/VLR		Supply installed defaults
CKUT	3	FLR	B3.3	Sortout check
COBU	3	FLR	B3.3	Exit E35 interface for invoked sort
COBV	1	VLR	B3.1	Exit E15 with no SORTIN interface
CYLN	1,2,3	FLR/VLR		Cylinder selection
DEVT	All	FLR/VLR		Device table
DYNA	0	FLR/VLR	D3.1	Dynamic allocation of SORTWK (MVS only)
EXII	1	FLR	B3.1	Phase 1 exit
EXIK	2	FLR	B3.2	Phase 2 exit
EXIN	0	FLR/VLR	D3.1	Phase 0 exit
EXIO	3	FLR	B3.3	SMF write and final exit from sort
EXIT	3	FLR	B3.3	Final exit point
E15B	1	FLR	B3.1	Exit E15 interface
E35B	3	FLR	B3.3	Exit E35 interface
FAUL	0	FLR	D3.1	Initialization values
FILI	1	FLR	B3.1	Load module filler
FILK	2	FLR	B3.2	Load module filler
FILO	3	FLR	B3.3	Load module filler
FORM	0	FLR	D3.1	Control field format tables
FVLI	1	VLR	B3.1	Input phase load module filler
FVLK	2	VLR	B3.2	Key Phase load module filler
FVLN	0	FLR/VLR	D3.1	Phase 0 load module filler
FVLO	3	VLR	B3.3	Output phase load module filler
GENV	1	VLR	B3.1	Point input code generation
INIB	0	FLR/VLR	D3.1	Control statement handling
INIC	0	FLR/VLR	D3.1	Generate extract/restore code
INID	0	FLR/VLR	D3.1	Initialize move and tree code
INIO	0	FLR/VLR	D3.1	Transfer conditions for other techn.
IPUB	1	FLR	B3.1	Phase 1 I/O handling
IPUM	1	FLR	B3.1	Handle records and indexes
IPUT	1	FLR	B3.1	Initialize for Phase 1
IPVB	1	VLR	B3.1	Input phase I/O handler
IPVM	1	VLR	B3.1	Handles input records and indexes
IPVT	1	VLR	B3.1	Initialize for input phase

FLR = fixed-length record
VLR = variable-length record

Module Name	Used in Phase:	FLR or VLR	MO Diag	Purpose
KPUA	2	FLR	B3.2	Virtual block construction
KPUB	2	FLR	B3.2	Phase 2 I/O handling
KPUS	2	FLR	B3.2	Virtual block construction
KPUT	2	FLR	B3.2	Initialize for Phase 2 or 3
KPUV	2	FLR	B3.2	Virtual block construction
KPVA	2	VLR	B3.2	Virtual block construction
KPVB	2	VLR	B3.2	Key phase I/O handler
KPVS	2	VLR	B3.2	Key phase writeback merge
KPVT	2	VLR	B3.2	Initialize for key and output phases
KPVV	2	VLR	B3.2	Virtual block construction
LIMI	3	FLR	B3.2	Writeback limit
MAN	0	FLR/VLR	D3.1	Handle parameter list and DD statements
MESI	1	FLR		Handle Phase 1 messages
MESS	All	FLR/VLR		Message handling
MESV	1	VLR		Handle input phase messages
MSGI	1	FLR/VLR		Phase 1 messages
MSGK	2	FLR/VLR		Phase 2 messages
MSGN	0	FLR/VLR		Phase 0 messages
MSGO	3	FLR/VLR		Phase 3 messages
OPUA	3	FLR		Phase 3 subroutines
OPUB	3	FLR	B3.3	Phase 3 I/O handling
OPUT	3	FLR	B3.3	Handle records and indexes
OPVA	3	VLR	B3.3	Output phase subroutines
OPVB	3	VLR	B3.3	Output phase I/O handler
OPVT	3	VLR	B3.3	Main output driver
OXIV	3	VLR	B3.3	Output exit and message entry points
OXOV	3	VLR	B3.3	SMF write and final exit from sort
SUBS	1,3	FLR		Suballocate main storage
SUBX	1,3	VLR		Suballocate main storage
XPUT	1,2,3	FLR		Load user exits
15V	1	VLR	B3.1	Exit E15 with SORTIN interface
35VL	3	VLR	B3.3	Exit E35 interface

FLR = fixed-length record
VLR = variable-length record

Peerage and Vale Directory

This directory lists Peerage and Vale load modules. Their names are in the form ICEmod, where mod is the three or four character identification of the module. Only these last characters are listed. The list also gives a reference to the Method of Operation diagrams (MO diag) provided in Section 2.

Module Name	Used in Phase:	MO Diag	Purpose
AM1	0		Supply installed defaults
BUG	All		Handle formatted dumps
CKP	2		Prepare for checkpoint and/or release unused work space
COM	All		Source code for ICECOMMA (formatted dump)
CRE	1	P3.1	Manage Phase 1
CRO	1		Final output in Phase 1
DEC	0	D3.2	
DED	0	D3.2	Manage Phase 0 (part 1)
DEF	0	D3.2	
DEG	0	D3.2, D3.3, P3.1	Manage Phase 0 (part 2)
DEV	0	D3.3	Manage Phase 0 (part 3)
ERR	0,1,3		Handle VSAM I/O errors
FIX (SVS/VS1)	0,1		Page fixing and freeing (SVC)
FIXO (OS)	0,1		DEB modification for I/O chaining (SVC)
LIM	3	P3.4	Manage Phase 3
LIV	3	P5.31	Handle VSAM output in Phase 3 (Peerage only)
MAN	0	D3.1	Handle transfer from Blockset
MON	All		Overall program management
OBS	0		Open for SORTIN/SORTOUT EXCP
PAR	2	P3.2	Manage Phase 2 (partitioning)
RED	2	P3.3	Manage Phase 2 (merging)
VED	2	P3.5	Manage Phase 2 (merging, with variable length or very long records)
VEE	2	P3.5	Take E25 if specified
VIM	3	P3.6	Manage Phase 3 (variable length or very long records)
VIP	3		Initialize Phase 3
VRE	1	P3.1	Manage Phase 1 (variable length and very long records)
VRN	1	P3.1	Manage Phase 1 (variable length or very long records)
VRO	1		Final output in Phase 1
XCP	0		Build channel programs for SORTIN and SORTOUT

Directory for Conventional Techniques

The names are listed alphabetically. All CSECTS are included, plus any other labels referenced in other sections of this publication.

EXPLANATION OF COLUMN HEADINGS

CSECT Name

Naming conventions are described in Appendix D.

Each CSECT name consists of six characters, the first three of which are always ICE. For easy reference, the names are listed by their last three characters only. In Phases 1, 2 and 3 those which begin with 'A' or '9' are generally assignment modules, and those which begin with 'R' or '8' are running modules. With a few exceptions, module Axy will carry out assignment for module Rxy, and so on.

Used in Phase:

The phases are:

0 (Def)	- Definition
0 (Opt)	- Optimization
1	- Sort (only for sorting applications)
2	- Intermediate merge (only for sorting applications)
3	- Final merge

The first and last phases have some alternative modules for sorting applications and for merge-only applications, which are indicated as follows:

3 Sort	- Used in Phase 3 of a sorting application
3 Merge	- Used in the merge phase of a merge-only application
3 All	- Can be used in this phase for all applications

The words 'sort', 'merge' and 'all' are used with 0 (Def) and 0 (Opt) in the same way.

MO Diagram

Provides a cross reference to the relevant diagram in the Method of Operation section (Section 2).

Purpose

The major purpose of the CSECT is briefly described. Keep in mind the notes on modules beginning with 'A' and '9' (above, under 'CSECT Name').

Mult	- Means a multiple compare routine is used to compare control fields;
No Mult	- Means it is not;
Extract	- Means an extract routine is used to compare control fields.

CSECT Name	Used in Phase:	MO Diag	Purpose
ABA	1		CRCX & OSCL: block var. records
ABB	1		Block fixed-length records with in-line move
ABC	1		Block fixed-length records with link to multiple move
ABE	1		Block var. records with move
ABF	1		Move var. records
ABG	2		Block or deblock fixed-length records with in-line move (no exits)
ABH	2		Block or deblock fixed-length records with link to multiple move (no exits)
ABI	2		Block or deblock var. records (no exits)
ABJ	2		Block or deblock fixed-length records (exits used)
ABK	2		Block or deblock var. records (exits used)
ABL	3 all		Block fixed-length records with in-line move (no exits)
ABM	3 all		Block fixed-length records (exits used)
ABN	3 all		Block var. records (no exits)
ABO	3 all		Block var. records (exits used)
ABP	3 all		Block fixed-length records with link to multiple move
ABQ	3 all		Move generator for fixed-length records
ABR	2	N3.2	Move generator for fixed-length records
ABS	1	N4.11	Move generator for fixed-length records
ABT	2		CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits)
ABU	2		CRCX & OSCL: block/deblock fixed-length records >256 bytes (no exits)
ABV	2		CRCX & OSCL: block/deblock var. records (no exits)
ABW	2		CRCX & OSCL: block/deblock fixed-length records (exits used)
ABX	2		CRCX & OSCL: block/deblock var. records (exits used)
ABY	1		CRCX & OSCL: block fixed-length records up to 256 bytes
ABZ	1		CRCX & OSCL: block fixed-length records >256 bytes
ADB	1		Deblock fixed-length records with in-line move (no exits)
ADC	1		Deblock fixed-length records with link to multiple move (no exits)
ADD	1		Deblock fixed-length records (exits used)
ADE	1		Deblock var. records (exits used)
ADG	1		Deblock var. records (no exits)
ADH	3 sort		Deblock fixed-length records
ADI	3 sort		Deblock var. records
ADJ	3 merge		Deblock (no exits)
ADL	2		Build tables in input buffer addresses
ADP	1		CRCX & OSCL: deblock fixed-length records up to 256 bytes (no exits)

CSECT Name	Used in Phase:	MO Diag	Purpose
ADQ	1		CRCX & OSCL: deblock fixed-length records >256 bytes (no exits)
ADR	1		CRCX & OSCL: deblock fixed-length records (exits used)
ADS	1		CRCX & OSCL: deblock var. records (exits used)
ADT	1		CRCX & OSCL: deblock var. records (no exits)
ADU	3 merge		Deblock (exits used)
ADX	3 sort		Deblock for read-forward tape, fixed-length records
AGA	1	N3.1	For tape, generate DCBs, IOBs, DCB addresses
AGB	2		Read tape backward
AGC	2		Read disk
AGD	3 sort		Read tape backward
AGE	3 sort		Read disk
AGF	3 merge		Open files
AGG	2	N3.2	For tape, generate DCBs, IOBs, DCB addresses
AGH	3 sort		Open files (and call checkpoint module)
AGI	1	N3.1	For disk, generate DCBs, IOBs, DCB addresses
AGJ	2	N3.2	For disk, generate DCBs, IOBs, DCB addresses
AGK	3 sort	N3.3	For disk, generate DCBs, IOBs, DCB addresses
AGL	2		For tape, read forward
AGM	3 sort		For tape, read forward
AGN	1	N3.1	OSCL: generate DCBs and IOBs
AMA	1		Contain messages for phase 1 assignment modules
AMB	2		Contain messages for phase 2 assignment modules
AMC	3 all		Contain messages for phase 3 assignment modules
AM1	0		Supply installed defaults
AOA	1		Sort fixed-length records, 'multiple' (not POLY)
AOB	1		Sort fixed-length records, 'single' (not POLY)
AOC	1		Sort var. records, 'multiple' (not POLY)
AOD	1		Sort var. records, 'single' (not POLY)
AOE	1		POLY: sort fixed-length records, 'multiple'
AOF	1		POLY: sort fixed-length records, 'single'
AOG	1		POLY: sort var. records, 'multiple'
AOH	1		POLY: sort var. records, 'single'
AOI	1		BALN (tape) sort algorithm
AOJ	1		POLY sort algorithm
AOK	1		BALN (disk) sort algorithm
AOL	0 (Opt) all	D3.4	Generate equals routine
AOM	0 (Opt) all	D3.4	Generate extract routine (part 1)
AON	1		OSCL sort algorithm
AOP	2 & 3 all		Merge, 'multiple'
AOQ	2 & 3 all		Merge, 'single'
AOR	2		BALN (tape) merge algorithm
AOS	2		POLY merge algorithm
AOT	2		BALN (disk) merge algorithm

CSECT Name	Used in Phase:		MO Diag	Purpose
AOW	1			CRCX & OSCL: initialization for fixed-length records, 'multiple'
AOX	1			CRCX & OSCL: initialization for fixed-length records, 'single'
AOY	1			CRCX & OSCL: initialization for var. records, 'multiple'
AOZ	1			CRCX & OSCL: initialization for var. records, 'single'
AO1	0	(Opt) sort		Optimize disk unit assignment
AO2	0	(Opt) sort		Optimize tape unit assignment
AO4	0	(Opt) all	D3.4	Generate extract routine (part 2)
AO5	0	(Opt) all	D3.4	Generate extract routine (part 3)
APA	1			Write tape
APB	1			Write disk
APC	1			Open files (and call checkpoint module); skip records on SORTIN
APD	2			Write tape
APE	2			Write disk
APF	3	merge	N3.3	Generate DCBs, DCB addresses
APG	1		N3.1	Allocate storage for I/O buffers, RSA, (three, control blocks, etc.)
APH	2		N3.2	Allocate storage for I/O buffers, control blocks, etc.
API	3	all	N3.3	Allocate storage for I/O buffers, control blocks, etc.
APJ	2			Open files (and call checkpoint module)
APK	3	sort	N3.3	For tape: generate DCBs, IOBs, DCB addresses
APL	1		N3.1	CRCX & OSCL: allocate storage for I/O buffers, RSA, control blocks, etc.
AP1	1			Specify area for DCB list for OPEN
AP2	2			Specify area for DCB list for OPEN
AP3	3			Specify area for DCB list for OPEN
BGB	0	(Def) sort	D3.4	CRCX: calculate B and G
CHK	1,2,3	sort		Checkpoint routine for sorting applications
DM4	All			Hexadecimal and decimal conversion
EX1	1			Act as link between user routines and program exits.
EX2	2			Routines which need linkage editing are linkage edited together with these modules:
EX3	3	all		phase 1 routines with EX1, and so on.
RBA	1		N4.11	CRCX & OSCL: block var. record
RBB	1		N4.11	Block fixed-length records with inline move
RBC	1		N4.11	Block fixed-length records with link to multiple move
RBE	1		N4.11	Block var. records with move
RBF	1		N4.11	Move var. records
RBG	2		N3.2	Block or deblock fixed-length records with in-line move (no exits)

CSECT Name	Used in Phase:	MO Diag	Purpose
RBH	2	N3.2	Block or deblock fixed-length records with link to multiple move (no exits)
RBI	2	N3.2	Block or deblock var. records with move (no exits)
RBJ	2	N3.2	Block or deblock fixed-length records (exits used)
RBK	2	N3.2	Block or deblock var. records (exits used)
RBL	3 all	N3.3	Block fixed-length records with in-line move (no exits)
RBM	3 all	N3.3	Block fixed-length records (exits used)
RBN	3 all	N3.3	Block var. records (no exits)
RBO	3 all	N3.3	Block var. records (exits used)
RBP	3 all	N3.3	Block fixed-length records with link to multiple move
RBT	2	N3.2	CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits)
RBU	2	N3.2	CRCX & OSCL: block/deblock fixed-length records >256 bytes (no exits)
RBV	2	N3.2	CRCX & OSCL: block/deblock var. records (no exits)
RBW	2	N3.2	CRCX & OSCL: block/deblock fixed-length records (exits used)
RBX	2	N3.2	CRCX & OSCL: block/deblock var. records (exits used)
RBY	1	N4.11	CRCX & OSCL: block fixed-length records up to 256 bytes
RBZ	1	N4.11	CRCX & OSCL: block fixed-length records >256 bytes
RCA	All but 0 (Def)		Specify PPI area
RCB	All but 0 (Def)		System interface
RCD	0 (Def) all	D3.4	Scan MODS control statement
RCI	0 (Def) all		Identify input, output and work data sets and their characteristics
RCJ	0 (Opt) sort		Check disk capacity
RCK	0 (Opt) sort	D3.4	BALN (disk): optimize B and G
RCL	0 (Def) merge	D3.4	Merge-only optimization
RCM	0 (Def) all	D3.4	System interface
RCN	0 (Def) sort	D3.4	Calculate bin sizes
RCO	0 (Def) all		System interface controlling phase 0 (Def) (and linkage editor)
RCP	0 (Def) all	D3.4	List exit routines to be linkage edited
RCQ	0 (Def) all		Specify input area for control statements
RCR	0 (Def) sort	D3.4	Calculate B and G for tape sort and choose
RCS		D3.4	optimal technique; interpret MODS statement
RCT	1,2,3 all		Free storage between phases
RCU	0 (Def+Opt) all		Contain error messages for phase 0
RCV	1,2,3 all		System interface for processing records
RCX	0 (Def) all		Contain messages for MODS card errors
RCZ	0 (Opt) all		Control of phase 0 (Opt)
RC1	0 (Opt) all	D3.4	Move CPI information to PPI
RC2	0 (Def) all		Calculation for extract
RC3	0 (Def) all		Contains messages for MODS card scanning

CSECT Name	Used in Phase:	MO Diag	Purpose
RC4	0 (Opt) sort		Calculate and optimize disk work areas
RC6	1	N3.1	Determine modules needed in phase 1
RC7	2	N3.2,N3.1	Determine modules needed in phase 2
RC8	3 all	N3.3	Determine modules needed in phase 3
RC9	1,2,3 all	N3.3, N3.2, N3.1	Load necessary modules for each phase
RDB	1	N3.1	Deblock fixed-length records with in-line move (no exits)
RDC	1	N3.1	Deblock fixed-length records with link to multiple move (no exits)
RDD	1	N3.1	Deblock fixed-length records (exits used)
RDE	1	N3.1	Deblock var. records (exits used)
RDG	1	N3.1	Deblock var. records (no exits)
RDH	3 sort	N3.3	Deblock fixed-length records (for disk and read-backward tape)
RDI	3 sort	N3.3	Deblock var. records
RDJ	3 merge	N3.3	Deblock
RDL	2		Set up deblock area
RDP	1	N3.1	CRCX & OSCL: deblock fixed-length records up to 256 bytes (no exits)
RDQ	1	N3.1	CRCX & OSCL: deblock fixed-length records up to 256 bytes (no exits)
RDR	1	N3.1	CRCX & OSCL: deblock fixed-length records (exits used)
RDS	1	N3.1	CRCX & OSCL: deblock variable records (exits used)
RDT	1	N3.1	CRCX & OSCL: deblock var. records (no exits)
RDU	3 merge	N3.3	Deblock (exits used)
RDX	3 sort	N3.3	Deblock fixed-length records (for read-forward tape)
RGB	2	N3.2	Read tape backwards
RGC	2	N3.2	Read disk
RGD	3 sort	N3.3	Read tape backwards
RGE	3 sort	N3.3	Read disk
RGF	3 merge		Indicate input end-of-file
RGL	2	N3.2	Read tape forward
RGM	3 sort	N3.3	Read tape forward
RGV	1 sort, 3 merge	N3.1, N3.3	VSAM read
RMA	1		Contain messages for ph1 running modules
RMB	2		Contain messages for ph2 running modules
RMC	3 all		Contain messages for ph3 running modules
ROA	1	N4.11	Sort fixed-length records, 'multiple' (not POLY)
ROB	1	N4.11	Sort fixed-length records, 'single' (not POLY)
ROC	1	N4.11	Sort var. records, 'multiple' (not POLY)
ROD	1	N4.11	Sort var. records, 'single' (not POLY)
ROE	1	N4.11	POLY: sort fixed-length records, 'multiple'
ROF	1	N4.11	POLY: sort fixed-length records, 'single'
ROG	1	N4.11	POLY: sort var. records, 'multiple'

CSECT Name	Used in Phase:	MO Diag	Purpose
ROH	1	N4.11	POLY: sort var. records, 'single'
ROI	1	N4.12	BALN (tape): sort algorithm
ROJ	1	N4.12	POLY: sort algorithm
ROK	1	N4.12	BALN (disk): sort algorithm
RON	1	N4.12, N3.2	OSCL: sort algorithm; initiate checkpoint operations
ROP	2 & 3 all	N3.2, N3.3	Merge records 'multiple'
ROQ	2 & 3 all	N3.2 N3.3	Merge records, 'single'
ROR	2	N3.2	BALN (tape): merge algorithm
ROS	2	N3.2	POLY: merge algorithm
ROT	2	N3.2	BALN (disk): merge algorithm
ROW	1		CRCX & OSCL: sort fixed-length records, 'multiple'
ROX	1		CRCX & OSCL: sort fixed-length records, 'single'
ROY	1		CRCX & OSCL: sort var. records, 'multiple'
ROZ	1		CRCX & OSCL: sort var. records, 'single'
RPA	1	N4.12, N3.2	Write tape
RPB	1	N4.12	Write disk
RPC	1	N3.1	End-of-phase housekeeping (not CRCX or OSCL)
RPD	2	N3.2	Write tape
RPE	2	N3.2	Write disk
RPF	2	N3.2	End-of-phase housekeeping (not CRCX or OSCL)
RPG	3 all	N3.3	Open output; end-of-phase housekeeping procedures
RPM	1 & 2	N3.2	OSCL: end-of-phase housekeeping
RPV	3 all	N3.3	VSAM write
8BN	3 all	N3.3	Block var. spanned records (no exits)
8BO	3 all	N3.3	Block var. spanned records (exits used)
8CI	0 (Def) all		Contain tables and constants for RCI
8CK	0 (Opt) sort		BALN (disk) timing estimate routine
8CR	3 merge		Checkpoint routine at end of output volume
8DJ	3 merge	N3.3	Deblock variable-length spanned records (no exits)
8GB	2	N3.2	CRCX read
8GC	3 sort	N3.3	CRCX read
8ON	1	N4.12, N3.2	CRCX sort algorithm
8PA	1 & 2	N4.12, N3.2	CRCX write
8PM	1 & 2	N3.2	CRCX end-of-phase housekeeping
9BN	3 all		Block var. spanned records, no exits (assignment)
9BO	3 all		Block var. spanned records, exits used (assignment)

CSECT Name	Used in Phase:	MO Diag	Purpose
9DJ	3 merge		Assignment deblock variable-length spanned records (no exits)
9GB	2		CRCX read assignment
9GC	3 sort		CRCX read assignment
9GN	1	N3.1	CRCX: generate DCBs, OBs, ECBs, alternate CCW pointers
9ON	1		CRCX algorithm assignment
9PA	1 & 2		CRCX write assignment

Section 5: Data Areas

The program makes extensive use of a large data area. For Blockset applications the area is called COMMON, and its layout is defined in the DSECT COMMON, for Peerage and Vale applications the area is called COMMA, and its layout is defined in the DSECT ICECOMMA. For other applications the information in COMMA is moved in phase 0 to a 'control phase information area', the CPI, and thence to a 'phase-to-phase information area', the PPI, the DSECT for which resides in module ICERCA. How to list COMMA is described in Section 6. The CPI and PPI are described in detail below together with Module ICEAM1, which contains CSECT ICEAM1, which contains the default values specified when the program was installed.

For the CPI and PPI a listing is given showing the position, length, name and contents of each field. For the PPI a 'map' is also provided, as well as an index.

If you know the name of a field in the CPI or the PPI, but not its position, you can find its displacement from the cross-reference tables in Section 6.

Blockset Area (COMMON)

This area is allocated by ICEMAN and is used by all Blockset modules. It is described in the DSECT COMMON. The program uses register 13 as a base register for COMMON.

COMMON PRINTOUT

```
DISPL NAME                                   DESCRIPTION
-------------------------------------------------------------------------------

      ********************************************************************************
0000  COMMON   DSECT
      *------------------------------------------------------------------------------*
      *              SAVE AREA                                                       *
      *------------------------------------------------------------------------------*
0000            DS    3F                     START OF OS SAVE AREA
000C  COMSAVE0 DS    15F                    LEVEL ZERO SAVE AREA
0048  COMSAVE1 DS    13F                    LEVEL ONE SAVE AREA
007C  COMSAVE2 DS    13F                    LEVEL TWO SAVE AREA
00B0  COMSAVE3 DS    13F                    LEVEL THREE SAVE AREA
      *------------------------------------------------------------------------------*
      *              FOUNDATION                                                      *
      *------------------------------------------------------------------------------*
00E4  COMFOUND DS    0F
      *
      *      PRESET DEFAULTS
      *
00E4  COMFAULT DS    0F
      *      VERSION NUMBER OF COMMON
      COMVERSE EQU   3                      CURRENT COMMON VERSION NUMBER
00E4  COMVERZN DS    H                      CONTAINS COMVERSE
      *      MODE INDICATORS
00E6  COMVOKED DS    C                      USER INVOKED STATUS
00E7  COMMODE  DS    C                      SORT OR MERGE MODE INDICATOR
00E8  COMCOBIN DS    C                      COBOL INPUT MODE OPTION
00E9  COMCOBUT DS    C                      COBOL OUTPUT MODE OPTION
00EA  COMRECFM DS    C                      SORT RECORD FORM
00EB  COMMNOGO DS    C                      ERROR ENCOUNTERED INDICATOR
00EC  COMABEND DS    C                      ABDUMP DD ENCOUNTERED
00ED  COMSYSTM DS    C                      SYSTEM TYPE FROM CVT
00EE  COMMTEST DS    C                      TEST STATUS INDICATOR
      COMMT190 EQU   128                         ISSUE 190 MESSAGES
      COMMTSEQ EQU   64                          SEQUENCE CHECK SORTOUT
      COMMSNAP EQU   32                          SNAP WRITEBACK BLOCKS
      COMMTVER EQU   32                          VERIFY BIN COUNT
      COMFLAG  EQU   16                          TRACE INPUT BLOCKS
      COMMTPH0 EQU   8                           ISSUE PHASE0 MSGS
00EF  COMINOUT DS    C                      COMMON IN/OUT DATA SET
00F0  COMIOTYP DS    C                      V = VIRTUAL, R = REAL
00F1  COMEOV   DS    C                      END OF VOLUME ON SORTIN INDICATOR
00F2  COMEOF   DS    C                      END OF FILE ON SORTIN INDICATOR
00F3  COMSKEYS DS    C                      SECONDARY KEYS PRODUCED
00F4  COMPMODE DS    C                      'L' = LOW, 'H' = HIGH.
00F5  COMSKIPM DS    C                      'Y' = SKIP INTERMEDIATE MERGE
      *      MESSAGE CSECT
00F8  COMDIAGS DS    F                      DIAGNOSTIC LIST ADDRESS
      *      CONSTANTS
00FC  COMINUS4 DS    F                      MINUS 4
0100  COMINUS8 DS    F                      MINUS 8
```

```
DISPL NAME                                    DESCRIPTION
------------------------------------------------------------------------

     *-------------------------------------------------------------------*
     *              USER SPECIFIED OPTIONS                               *
     *-------------------------------------------------------------------*
0104 COMOPTNS DS    0F
     *      ALTERNATE SORT NAME
0104 COMMXCTL DS    CL8                      NAME TO XCTL TO
     *      WTO OPTIONS
010C COMOPWTO DS    CL12                     WTO HEADER AND ROUTING
     *      STORAGE OPTIONS
0118 COMAXLIM DS    F                        MAX FOR SIZE=MAX
011C COMMAIN  DS    F                        MAX FOR SIZE= NOT SPECIFIED
0120 COMSERVE DS    F                        AMOUNT OF RESERVED MEMORY
     *      DDNAME OPTIONS
0124 COMSPFIX DS    CL4                      'SORT' DD PREFIX OR ALIAS
0128 COMSYSUT DS    CL8                      SYSOUT FILE NAME
0130 COMCSYSO DS    CL8                      COBOL SYSOUT=A DDNAME
     *      SORTIN/SORTOUT MEMBER NAMES
     COMEMBRI EQU   COMSYSUT                 SORTIN MEMBER
     COMEMBRO EQU   COMCSYSO                 SORTOUT MEMBER
     *      MESSAGE CLASS OPTIONS
0138 COMPRINT DS    C                        PRINT MESSAGE CLASS
0139 COMCPRNT DS    C                        COBOL SORT MESSAGE PRINT CLASS
013A COMTYPE  DS    C                        TYPE MESSAGE CLASS
013B COMLOG   DS    C                        LOG MESSAGE CLASS
     *      LISTING CONTROL COUNTS
013C COMLCTR  DS    C                        INITIAL LINE COUNTER VALUE
013D COMLINES DS    C                        LINES PER PAGE
     *      CONTROL CARD LIST OPTION
013E COMSGLST DS    C                        LIST CONTROL CARDS OPTION
     *      MESSAGE TEXT OPTIONS
013F COMPFIX  DS    CL3                      PREFIX NAME
0142 COMARKER DS    C                        CONTROL CARD ERROR POSTION FLAG
0143 COMJOB   DS    C                        TYPE/LOG JOB NAME
0144 COMSTEP  DS    C                        TYPE/LOG STEP NAME
     *      ABNORMAL TERMINATION OPTIONS
0145 COMIBEND DS    C                        ABEND IF ERROR DURING INITIAL PHASE
0146 COMIDUMP DS    C                        DUMP OPTION FOR IBEND
0147 COMOBEND DS    C                        ABEND IF ERROR DURING OTHER PHASE
0148 COMODUMP DS    C                        DUMP OPTION FOR OBEND
     *      LINK PACK AREA MODE INDICATOR
0149 COMLPA   DS    C                        IPUT AND OPUT ARE IN LPA
     *      CONSORT SVC
014A COMSVC   DS    H                        SVC FOR SORT
     COMOPTZ  EQU   *-COMOPTNS               SIZE OF OPTIONS
     *-------------------------------------------------------------------*
     *              DCB POINTERS                                         *
     *-------------------------------------------------------------------*
014C COMPRDCB DS    F                        ADDRESS OF SYSOUT DCB
0150 COMRDDCB DS    F                        ADDRESS OF SYSIN DCB
0154 COMINDCB DS    F                        SORTIN DCB ADDRESS
0158 COMUTDCB DS    F                        SORTOUT DCB ADDRESS
015C COMWKDCB DS    F                        SORTWK OPEN LIST ADDRESS
0160 COM01DCB DS    F                        SORTIN01 OPEN LIST ADDRESS
     *    UNIT RECORD DECBS
0164 COMPRDEC DS    F                        ADDRESS OF SYSOUT DEC
0168 COMRDDEC DS    F                        SYSIN DECB ADDRESS
     *-------------------------------------------------------------------*
     *              ADDRESSES OF VARIOUS GENERATED ROUTINES IN COMMON     *
     *-------------------------------------------------------------------*
016C COMLOAD  DS    F                        LOAD SUBROUTINE ADDRESS
0170 COMUNLD  DS    F                        UNLOAD SUBROUTINE ADDRESS
0174 COMKLOAD DS    F                        KEY LOAD SUBROUTINE ADDRESS
0178 COMKUNLD DS    F                        KEY UNLOAD SUBROUTINE ADDRESS
017C COMOVREC DS    F                        ADDRESS OF RECORD MOVE ROUTINE
0180 COMLDKAD DS    F                        ADDRESS OF KAD LOAD ROUTINE
0184 COMSTORE DS    F                        RESTORE CODE ADDRESS
```

```
DISPL NAME                                    DESCRIPTION
------------------------------------------------------------------------

     *-------------------------------------------------------------------*
     *              SORTIN/SORTOUT JFCB FIELDS                           *
     *-------------------------------------------------------------------*
0188 COMIFORM DS    C                        INPUT RECFM
0189 COMOFORM DS    C                        OUTPUT RECFM
018A COMIBLKZ DS    H                        INPUT BLOCKSIZE
018C COMOBLKZ DS    H                        OUTPUT BLOCKSIZE
018E COMIRECZ DS    H                        INPUT LRECL
0190 COMORECZ DS    H                        OUTPUT LRECL
0192 COMIBLKL DS    H                        INPUT FILE AVERAGE BLOCK LENGTH
0194 COMVOLCT DS    C                        SORTIN VOLUME COUNT
     *      OTHER SORTIO FIELDS
0196 COMINIO  DS    H                        MINIMUM SORTIO BUF TRANSFER COUNT
0198 COMAXIO  DS    H                        MAXIMUM SORTIO BUF TRANSFER COUNT


     *-------------------------------------------------------------------*
     *              RECORD COUNTS                                        *
     *-------------------------------------------------------------------*
019C COMFILEZ DS    F                        FILE SIZE
01A0 COMIRECS DS    F                        INPUT REC COUNT
01A4 COMORECS DS    F                        OUTPUT REC COUNT
01A8 COMARECS DS    F                        ADDED REC COUNT
01AC COMDRECS DS    F                        DELETED REC COUNT
01B0 COMRUNSZ DS    F                        CURRENT RUN LENGTH
01B4 COMVALCT DS    F                        BVAL STANDARD COUNT
01B8 COMVALPR DS    F                        BVAL PRIOR COUNT
     *-------------------------------------------------------------------*
     *              OTHER COUNTS                                         *
     *-------------------------------------------------------------------*
01BC COMMAIN0 DS    0F                       SAVED INIT MAIN STORAGE VALUE
01BC COMFREEB DS    F                        FREE BIN COUNT
01C0 COMMAIN1 DS    0F                       SAVED MAIN STORAGE VALUE
01C0 COMBINSN DS    F                        SELECTED BIN RECORD COUNT
01C4 COMRUNS  DS    F                        RUNS DISTRIBUTED BY INPUT PHASE
01C8 COMMAIN2 DS    0F                       USED MAIN STORAGE VALUE
01C8 COMRSASZ DS    F                        RSA SIZE
01CC COMIRSAZ DS    F                        INPUT RSA SIZE
01D0 COMKOUNT DS    F                        PRIMARY KAD COUNT
     *-------------------------------------------------------------------*
     *              MEMORY MANAGEMENT FIELDS                             *
     *-------------------------------------------------------------------*
     *      FILE AREA POINTERS
01D4 COMFILEG DS    F                        GET FILE AREA PTR
01D8 COMFILEP DS    F                        PUT FILE AREA PTR
     *      MAIN STORAGE FIELDS
01DC COMMAREA DS    F                        ADDRESS OF COMMON WORK AREA
01E0 COMMEND  DS    F                        ADDRESS OF END OF COMMON WORK AREA
01E4 COMAVAIL DS    F                        ADDRESS OF AVAILABLE COMMON
01E8 COMSAD   DS    F                        ADDRESS OF FIRST STORAGE AREA DESC
01EC COMSPACE DS    F                        BYTES AVAILABLE
01F0 COMINGET DS    F                        MINIMUM GETMAIN AMOUNT
01F4 COMINCOR DS    F                        MINIMUM SPACE NEEDED TO SORT
01F8 COMSHORT DS    F                        MAIN STORAGE DEFICIT
     *-------------------------------------------------------------------*
     *              COMPARE STRUCTURE DATA                               *
     *-------------------------------------------------------------------*
01FC COMLISTA DS    F                        LIST HEADERS
0200 COMLISTB DS    F                         FOR MERGE SORT.
0204 COMTREE  DS    F                        LOSER TREE AREA
0208 COMTSIZE DS    F                        OFFSET TO LAST NODE
```

```
DISPL NAME                                  DESCRIPTION
------------------------------------------------------------------------

     *-------------------------------------------------------------------*
     *              WORK BUFFER VALUES                                   *
     *-------------------------------------------------------------------*
020C COMBLKSI DS    F                      WORKFILE BLOCK SIZE
0210 COMRPB   DS    F                      RECORDS PER WORKFILE BLOCK
0214 COMKPB   DS    F                      KEYS PER WORKFILE BLOCK
     *      KAD VALUES
0218 COMKPFIX DS    F                      SIZE OF BIN KAD PREFIX
021C COMKDATA DS    F                      SIZE OF KAD PREFIX
0220 COMKBLKZ DS    F                      SIZE OF KAD BLOCK ENTRY
     *-------------------------------------------------------------------*
     *              I/O TRANSFER AREA FIELDS                             *
     *-------------------------------------------------------------------*
0224 COMIOXAD DS    F                      ADDRESS OF IOX
0228 COMIOXSZ DS    F                      LENGTH OF IOX
022C COMLOWIO DS    F                      LOW I/O ADDRESS
0230 COMIOLOC DS    F                      SORTIO DATA ADDRESS IN IOX
0234 COMWKLOC DS    F                      SORTWK DATA ADDRESS IN IOX


     *-------------------------------------------------------------------*
     *              DXL LISTS                                            *
     *-------------------------------------------------------------------*
0238 COMDXLSQ DS    F                      SEQUENTIAL DXL LIST HEAD
023C COMDXLIO DS    F                      SORTIO CIRCULAR LIST ENTRY
0240 COMDXLWQ DS    F                      SORTWK DXL WRITE QUEUE
0244 COMDXLRQ DS    F                      SORTWK DXL READ QUEUE
0248 COMDXKEY DS    F                      ADDRESS OF KEY AREA DXL
024C COMDXSKY DS    F                      SECONDARY KEY AREA DXL ADDRESS
     *-------------------------------------------------------------------*
     *              QUEUES                                               *
     *-------------------------------------------------------------------*
0250 COMAREAQ DS    F                      WORK AREA PREFERENCE QUEUE
0254 COMBINFQ DS    F                      QUEUE OF AVAILABLE RECORD BINS
0258 COMLADFQ DS    F                      LAD AVAILABLE LIST
025C COMLADNQ DS    F                      NEXT STRING LAD QUEUE
0260 COMIOBXQ DS    F                      QUEUE OF EXCP IOBS
0264 COMBINSQ DS    F                      SELECTED BIN QUEUE
0268 COMBINST DS    F                      SELECTED QUEUE TAIL
026C COMKEYFQ DS    F                      FREE KEY HOLDER QUEUE
0270 COMCADFQ DS    F                      FREE CAD QUEUE
0274 COMCADAQ DS    F                      AVAILABLE CYLINDER AREA QUEUE
0278 COMBVDFQ DS    F                      FREE BVD QUEUE
027C COMBVDAQ DS    F                      ALLOCATED BVD QUEUE
0280 COMBVDLO DS    F                      ADDRESS OF LOW BOUNDARY VALUE DEF
0284 COMBVDHI DS    F                      ADDRESS OF HIGH BOUNDARY VALUE DEF
0288 COMKEYPQ DS    F                      PRIMARY KEY QUEUE
028C COMKEYPT DS    F                      PRIMARY KEY QUEUE TAIL
0290 COMKEYSQ DS    F                      SECONDARY KEY QUEUE
0294 COMKIRRF DS    F                      KAD IRRR HOLDER FREE QUEUE
0298 COMKIRRQ DS    F                      KAD IRRR HOLDER QUEUE
     *-------------------------------------------------------------------*
     *              KEY MISCELLANEOUS                                    *
     *-------------------------------------------------------------------*
029C COMOVALT DS    F                      ALTERNATE REC MOVE
02A0 COMTRACT DS    F                      EXTRACT ADDRESS
02A4 COMEQUAL DS    F                      EQUAL COUNTER
02A8 COMEXITS DS    F                      EXIT LIST HEAD
     *-------------------------------------------------------------------*
     *              VARIOUS FULLWORD SIZES                               *
     *-------------------------------------------------------------------*
02AC COMBINSZ DS    F                      SIZE OF A RECORD BIN
02B0 COMWRECZ DS    F                      SIZE OF A WORKFILE RECORD
02B4 COMDXLSZ DS    F                      SIZE OF DXL (DEPENDS ON V OR R)
02B8 COMPAREZ DS    F                      LENGTH OF COMPARE
02BC COMIORPB DS    F                      SORTIO REC PER BLOCK
```

```
DISPL NAME                                    DESCRIPTION
-------------------------------------------------------------------------------

      *-------------------------------------------------------------------------*
      *              TIOT ADDRESSES                                             *
      *-------------------------------------------------------------------------*
02C0 COMTIOT  DS    F                        ADDRESS OF TIOT
02C4 COMSAVEI DS    F                        TIOT ENTRY FORT SORTIN
02C8 COMSAVEO DS    F                        TIOT ENTRY FOR SORTOUT
      *       DISK SORTIO FIELDS
02CC COMCCHH  DS    F                        LATEST SORTIO CCHH
02D0 COMRECNO DS    2C                       LATEST R
02D2 COMTRBAL DS    H                        CURRENT TRACK BALANCE
02D4 COMTRSAV DS    H                        PREVIOUS TRACK BALANCE
02D6 COMXTENT DS    H                        CURRENT EXTENT
      *-------------------------------------------------------------------------*
      *              WTO AREA                                                   *
      *-------------------------------------------------------------------------*
02D8 COMWTO   DS    F                        HEADER FOR L FORM OF WTO
02DC COMWTEXT DS    CL120                    TEXT OF WTO
     COMWPFIX EQU   COMWTEXT                 MESSAGE PREFIX
     COMWMNUM EQU   COMWPFIX+3               MESSAGE NUMBER


     COMWMTAG EQU   COMWMNUM+3               MESSAGE TAG
     COMWJOB  EQU   COMWMTAG+2               JOB NAME
     COMWSTEP EQU   COMWJOB+9                STEP NAME
     COMWMESS EQU   COMWSTEP+9               MESSAGE
0354 COMWROUT DS    F                        ROUTING FOR WTO
      *-------------------------------------------------------------------------*
      *              MISCELLANEOUS                                              *
      *-------------------------------------------------------------------------*
0358 COMDUBBL DS    D                        DOUBLEWORD ALIGNED WORK AREA
0360 COMOLIST DS    F                        OPEN LIST ADDRESS
0364 COMOLEND DS    F                        END OF OPEN LIST
0368 COMSQRT  DS    F                        SQUARE ROOT OF RPB
036C COMIDKEY DS    F                        MIDDLE KEY
      *       MISCELLANEOUS BIT SWITCHES
0370 COMISCEL DS    4C                       SWITCH VALUES
     COMISREL EQU   1                        0 => RELEASE SORTWORK SPACE
     COMXSTAT EQU   COMISCEL+1               EXIT STATUS
     COMXE15  EQU   1                        E15 IS PRESENT
     COMXE35  EQU   2                        E35 IS PRESENT
     COMXE01  EQU   4                        E01 IS PRESENT
     COMXE03  EQU   16                       E03 IS PRESENT
      *       MORE MISCELLANEOUS
0374 COMBVD   DS    F                        CURRENT BVD
0378 COMLADS  DS    F                        LAD AREA ADDRESS
037C COMLADLM DS    F                        LAD AREA LIMIT
0380 COMIIOB  DS    F                        INPUT IOB
0384 COMOIOB  DS    F                        OUTPUT IOB
0388 COMACE   DS    F                        ABNORMAL CHANNEL END
038C COMSIRRR DS    F                        SECONDARY KEYFILE IRRR
0390 COMBOUND DS    F                        COUNT OF WITHIN BOUND RECORDS
0394 COMLIMIT DS    F                        BOUND LIMIT
      *-------------------------------------------------------------------------*
      *              KEYFILE COUNTS AND LIMITS                                  *
      *-------------------------------------------------------------------------*
0398 COMKEYPC DS    H                        PRIMARY KEY COUNT
039A COMKEYSC DS    H                        SECONDARY KEY COUNT
039C COMKEYPN DS    H                        PRIMARY KEY MINIMUM
039E COMKEYPX DS    H                        PRIMARY KEY MAXIMUM
03A0 COMKEYSX DS    H                        SECONDARY KEY MAXIMUM
03A2 COMINKPB DS    H                        MINIMUM KEYS PER BLOCK
03A4 COMAXKPB DS    H                        MAXIMUM KEYS PER BLOCK
```

```
DISPL NAME                                DESCRIPTION
-------------------------------------------------------------------------------

     *-------------------------------------------------------------------*
     *              MISCELLANEOUS HALFWORD FIELDS                        *
     *-------------------------------------------------------------------*
03A6 COMKRECZ DS    H                    KEY RECORD LENGTH
03A8 COMINBOF DS    H                    BUFFER OFFSET FOR ASCII SORTIN
03AA COMKDELT DS    H                    PRIMARY KEY MULTIPLE LOAD DELTA
03AC COMLSTAR DS    H                    1ST TWO BYTES OF JFCLSTAR
03AE COMWDXLS DS    H                    WORKFILE DXLS
03B0 COMCYLNO DS    H                    LOGICAL CYLINDER ASSIGNMENT CTR
03B2 COMPRIZE DS    H                    PRIOR BLOCK SIZE
03B4 COMBVCTR DS    H                    HIGH BV XFER COUNT
03B6 COMBVLIM DS    H                    HIGH BV XFER LIMIT
     *-------------------------------------------------------------------*
     *              KEY MOVES AND COMPARES - SEE TABLE IN KEYMVCLC       *
     *-------------------------------------------------------------------*
03B8 COMPMOVE DS    0H
03B8 COMXCRUN DS    CL6                  XC    0(,R1),0(R1)
03BE COMOVKEY DS    CL6                  MVC   0(,R1),0(R15)
03C4 COMPAB   DS    CL6                  CLC   8(,R4),0(R3)
03CA COMPCOPA DS    CL6                  CLC   4(,R4),0(R14)
03D0 COMPCOPB DS    CL6                  CLC   4(,R5),0(R14)
03D6 COMPCOPC DS    CL6                  CLC   0(,R4),0(R14)
03DC COMPOUND DS    CL6                  CLC   0(,R2),0(R8)
03E2 COMPICK  DS    CL6                  CLC   0(,R2),BVDKEY-BVDSECT(R4)


03E8 COMPARE  DS    CL6                  CLC   0(,R14),0(R15)
03EE COMPSLAB DS    CL6                  CLC   4(,R4),4(R5)
     COMPSLAK EQU   COMPCOPA
     COMPSLBK EQU   COMPCOPB
03F4 COMPALTA DS    CL6                  CLC   4(,R2),0(R4)
03FA COMPALTB DS    CL6                  CLC   4(,R3),0(R4)
0400 COMOVKAD DS    CL18                 MVC   0(KADLIST,R1),0(R15)
     *-------------------------------------------------------------------*
     *              BLOCK READ CHECK FIELDS                              *
     *-------------------------------------------------------------------*
0418          DS    0D
0418 COMBLOKO DS    F                    OUTPUT BLOCK COUNTER
041C COMBLOKI DS    F                    INPUT BLOCK COUNTER
0420 COMHASH  DS    F                    SUM OF OUTPUT COUNT IDS
0424 COM2NDRY DS    F                    SECONDARY ALLOCATION WORD
0428 COMNMAX  DS    F                    ESTIMATED NMAX VALUE
     *-------------------------------------------------------------------*
     *              PHASE LOCAL WORK AREA                                *
     *-------------------------------------------------------------------*
042C COMLOCAL DS    22F
     *
0484 COMCLOSE DS    0F                   CLOSE LIST AREA
     COMIDSNM EQU   COMCLOSE             INPUT DATA SET NAME
     *-------------------------------------------------------------------*
     *              TREE/MERGE ROUTINE SPACE                             *
     *-------------------------------------------------------------------*
     *      MERGE/SORT INITIALIZATION ROUTINE
0484 COMERGEI DS    28F
     *      MERGE/SORT ROUTINE
04F4 COMERGES DS    48F
     COMTABLE EQU   COMERGES-4           TRANSLATE/SIZE TABLE
     *      TREE INITIALIZATION ROUTINE
05B4 COMTREEI DS    36F
     *      TREE ADD ROUTINE
0644 COMTREEA DS    40F
     *      TREE DELETE ROUTINE
06E4 COMTREED DS    16F
     *      TREE REPLACE/SELECT ROUTINE
0724 COM43LST DS    0F                   LIST AREA FOR CON043A MSGS
```

```
DISPL NAME                                    DESCRIPTION
------------------------------------------------------------------------

0724 COMTREES DS    20F
     *       COMPARE INSTRUCTION LIST
0774 COMPLIST DS    30F
07EC COMPLAST DS    F                         LAST LIST ENTRY
     *       LONG COMPARE
07F0 COMPLONG DS    0CL6                      LONG COMPARE
     *       OFFSET LIST TO LAD FETCHES IN TREE CODE
07F0 COMPLADS DS    6F
     *-------------------------------------------------------------------*
     *               VARIOUS WORK VARIABLES                              *
     *-------------------------------------------------------------------*
     *       DYNAMIC ALLOCATION PARAMETERS
0808           ORG   COMTABLE+256             SKIP OVER TABLE
05F0 COMDYNAM DS    CL8                       DYNALLOC UNIT NAME
05F8 COMDYNUM DS    F                         DYNALLOC COUNT
05FC COMVIO   DS    C                         VIO/NOVIO SWITCH
     *       ALTSEQ TRANSLATION TABLE
05FD COMALTAB DS    CL256
06FD           ORG
     *       E35 WORK AREAS
0808 COMOJFCB DS    F                         ADDRESS OF SORTOUT JFCB
080C COMLADAD DS    F                         BLOCKED LAD ADDR/ADDR(E35)
0810 COMRTAIL DS    F                         LAST FOUR BYTES OF RECORD
     *       USED FROM ICEDYNA UNTIL ECEEXIN



0814 COMDYNER DS    F                         RETURN CODE FROM DYNALLOC
     *       SYSTEM RESERVED
0818 COMSYSRV DS    F                         SYSTEM RESERVED
081C COMTRANS DS    X                         TRANSFER INDICATION FOR ICEMESS
081D COMCHALT DS    CL1                       A=TRANSLATE AQ ONLY
     *                                        C=TRANSLATE CH AND AQ              *
081E COMCHECK DS    CL1                       Y=RECORD COUNTERS CHECKED
     *                                        N=RECORD COUNTERS NOT CHECKED      *
081F COMSMF   DS    C                         'S' = SHORT SMF RECORD
     *                                        'F' = FULL SMF RECORD
     *                                        'F' NOT VALID FOR FLR-BLOCKSET
     *                                        'N' = NO SMF RECORD
0820 COMMXBLK DS    F                         MAX SORTIN BLKSIZE FOR SMF
0824 COMTTIME DS    F                         CPU TIME WORK AREA FOR SMF
0828 COMMSGSW DS    X                         INFORMATION MSG FLAG BYTE
     COMB37R  EQU   X'80'                    B37 INFO MESSAGE FLAG MASK
0829 COMVBLKS DS    C                         N=BYPASS VLR BLOCKSET
082A COMBLKS  DS    C                         N=BYBASS FLR BLOCKSET
082B COMCNTL  DS    C                         X'00'=SORTCNTL NOT PRESENT
082C COMTRACK DS    F                         #TRACKS ALLOCATED FOR SORTWK
0830 COMAVGLN DS    F                         AVERAGE RECORD LENGTH VALUE
     *-------------------------------------------------------------------*
     *               TRACE SEGMENTS                                      *
     *-------------------------------------------------------------------*
0834 COMTRACE DS    1F  TRACE X'01'-X'0F' E15/E35 IN/OUT AREA RECOR LEVEL
0838          DS    8F  TRACE X'10'-X'1F' BLOCK LEVEL ENTRIES
0858          DS    8F  TRACE X'20'-X'9F' BLOCKSET LEVEL ENTRIES
0878          DS   15F  TRACE X'A0'-X'FF' MODULE AND MAIN LOOPS
08B4           ORG   COMTRACE
0834 COMOPLOC DS    20F                       OPTION STATEMENT WORK AREA
0884           ORG   ,
     *-------------------------------------------------------------------*
     *               MORE SAVE AREAS                                     *
     *-------------------------------------------------------------------*
08B4 COMRSAV1 DS    F                         REGISTER SAVE AREA
08B8 COMRSAV2 DS    F                         REGISTER AREA
08BC COMMSGPR DS    H                         MESSAGE PARAMETER SAVE AREA
```

```
DISPL NAME                                    DESCRIPTION

      *-------------------------------------------------------------------*
      *              VLR FIELD AREA                                       *
      *-------------------------------------------------------------------*
08C0 COMIRES  DS    F                         START OF RESIDUAL INPUT DATA
08C4 COMWRES  DS    F                         START OF RESIDUAL WORK DATA
08C8 COMORES  DS    F                         START OF RESIDUAL OUTPUT DATA
08CC COMWKSIZ DS    F                         SIZE OF WORK AREA
08D0 COMFLOOR DS    F                         PAGE SIZE COMPLEMENT
08D4 COMDELTA DS    F                         RESIDUE DELTA
08D8 COMINSEG DS    F                         MIN SIZE FOR 1ST SEGMENT
08DC COMWKRSA DS    F                         WORK BUF TO RSA ROUTINE
08E0 COMRSAWK DS    F                         RSA TO WORK BUF ROUTINE
08E4 COMRSTOR DS    F                         RESTORE LIST POINTER
08E8 COMFSTOR DS    F                         FIXED RESTORE ROUTINE ADDRESS
08EC COMCENT  DS    F                         PER CENT RSA UTILIZATION
08F0 COMUNBAL DS    F                         ADDRESS OF UNBALANCED MERGE
08F4 COMLIST  DS    F                         UNBALANCED LIST HEAD
08F8 COMLEND  DS    F                         UNBALANCED LIST END
08FC COMPEERS DS    F                         PEERAGE ACCUMULATOR
0900 COMLOKEY DS    F                         POINTER TO LOW LIMIT KEY AREA
0904 COMTOTAL DS    F                         TOTAL RSA BYTES NOT IN OUTPUT RUN
0908 COMPARZ6 DS    F                         OFFSET IN BIN TO SEG BYTE
090C COMBACK  DS    F                         BACKOUT RECORD'S PREDECESSOR
0910 COMSGTOT DS    F                         TOTAL BYTES SEGMENTED
0914 COMWKFIL DS    F                         BYTES IN WORK FILE
0918 COMINFIL DS    F                         SORTIN BYTE COUNT
091C COMUTFIL DS    F                         SORTOUT BYTE COUNT


0920 COMESAVE DS    3F                        EVENT COUNT SAVE AREA
092C COMECTRS DS    10F                       EVENT COUNT SAVE AREA
0954 COMKSAVE DS    F                         KEY SAVE AREA ADDRESS
0958 COMLSTAT DS    F                         LENGTH STATISTICS CTR PTR
095C COMLSTAB DS    F                         LENGTH STATISTICS TABLE
0960 COMEXTRA DS    2F                        UNUSED WORDS
      *
0968 COMFEND  DS    0D                        END OF FOUNDATION
     COMMVARY EQU   *                         BEGINNING OF VARIABLE AREA
              END
```

Peerage and Vale Communication Area (COMMA)

This area is allocated by ICEDEF and is used by all Peerage and Vale technique modules. It is described in the DSECT ICECOMMA. The program uses register 13 as a base register for COMMA.

It can be listed as described in Section 6, 'Debugging Aids'. Such a listing gives not only field definitions (name, position, length, description), but also current contents.

DISP	LEVEL	LABEL	ATTR	COMMENT
000	2	CSAVEOS	PTR(31)	SUPERVISOR AND DM SAVE AREA
048	2	CSAVEL1		LEVEL 1 ROUTINE SAVE AREA
090	2	CSAVEL2	PTR(31)	LEVEL 2 ROUTINE SAVE AREA
0D8	2	CSAVEL3		LEVEL 3 ROUTINE SAVE AREA
120	2	CTEMP1	FIXED(31)	WORK AREA
120	3	CWORK1	FIXED(31)	WORK AREA
121	4	CTEMP124	PTR(24)	WORK AREA
121	5	CWORK124	PTR(24)	WORK AREA
122	6	CTEMP115	FIXED(15)	WORK AREA
122	7	CWORK116	FIXED(16)	WORK AREA
123	8	CTEMP108	PTR(8)	WORK AREA
123	9	CWORK108	PTR(8)	WORK AREA
124	2	CTEMP2	FIXED(31)	WORK AREA
124	3	CWORK2	FIXED(31)	WORK AREA
125	4	CTEMP224	PTR(24)	WORK AREA
125	5	CWORK224	PTR(24)	WORK AREA
126	6	CTEMP215	FIXED(15)	WORK AREA
126	7	CWORK216	FIXED(16)	WORK AREA
127	8	CTEMP208	PTR(8)	WORK AREA
127	9	CWORK208	PTR(8)	WORK AREA
128	2	CTEMP3	FIXED(31)	WORK AREA
128	3	CWORK3	FIXED(31)	WORK AREA
129	5	CTEMP324	PTR(24)	WORK AREA
129	6	CWORK324	PTR(24)	WORK AREA
12A	7	CTEMP315	FIXED(15)	WORK AREA
12A	8	CWORK316	FIXED(16)	WORK AREA
12B	9	CTEMP308	PTR(8)	WORK AREA
12B	10	CWORK308	PTR(8)	WORK AREA
12C	2	CTEMP4	FIXED(31)	WORK AREA
12C	3	CWORK4	FIXED(31)	WORK AREA
12D	4	CTEMP424	PTR(24)	WORK AREA
12D	5	CWORK424	PTR(24)	WORK AREA
12E	6	CTEMP415	FIXED(15)	WORK AREA
12E	7	CWORK416	FIXED(16)	WORK AREA
12F	8	CTEMP408	PTR(8)	WORK AREA
12F	9	CWORK408	PTR(8)	WORK AREA
130	2	CWORK5	FIXED(31)	WORK AREA
134	2	CWORK6	FIXED(31)	WORK AREA
138	2	CWORK7	FIXED(31)	WORK AREA
13C	2	CWORK8	FIXED(31)	WORK AREA
140	2	CRTNADDR		ADDR. OF PERMANENT RTNS
140	3	CGETMAIN	PTR(31)	ADDR. OF ROUTINE GETMAINS
144	3	CFRESTOR	PTR(31)	ADDR. OF ROUTINE FREESTOR
148	3	CGALADDR	PTR(31)	ADDR. OF ROUTINE GALLOCAT
14C	3	CRALADDR	PTR(31)	ADDR. OF ROUTINE RALLOCAT
150	3	CDALADDR	PTR(31)	ADDR. OF ROUTINE DALLOCAT
154	3	CBALADDR	PTR(31)	ADDR. OF ROUTINE BALLOCAT
158	3	CTRKROUT	PTR(31)	ADDR. OF ROUTINE TRKROUT
15C	3	CIORADDR	PTR(31)	ADDR. OF ROUTINE INOUTRTN
160	3	CMSGADDR	PTR(31)	ADDR. OF ROUTINE DIAGNOSE
164	3	CTIME	PTR(31)	ADDR. OF ROUTINE TIMING
168	3	CRECHAIN	PTR(31)	ADDR. OF ROUTINE RECHAIN
16C	3	CSTAADDR	PTR(31)	ADDR OF INIT STAE

DISP	LEVEL	LABEL	ATTR	COMMENT
170	2	CXLIST	PTR(31)	ADDRESS OF EXIT LIST
174	2	COMLOCSV	FIXED(31)	COREPARM SAVE FOR ICEDEV
178	2	COMMDBLE	BDY(DWORD)	DOUBLEWORD-ALIGNED WORKAREA
178	3	CXCCOUNT	FIXED(31)	WORK AREA FOR EXTRACT
178	4	CMOVEEND	PTR(31)	WORK AREA FOR MOVECOMP
178	5	CEQDISPL	PTR(31)	WORK AREA FOR EXTRACT
17C	3	CXCLNG	FIXED(31)	WORK AREA FOR EXTRACT
17C	4	CMOVEH	PTR(31)	WORK AREA FOR MOVECOMP
180	2	CFRELIST	BDY(DWORD)	START OF FREE SPACE CHAIN
180	3	CFREHEAD	PTR(31)	ADDR. OF FIRST FREE AREA
184	3	CFREZERO	FIXED(31)	SIZE OF ZEROTH FREE AREA
188	2	CADD	PTR(31)	ADDR. OF TREEADD ROUTINE
188	3	CADDISP	FIXED(31)	TREEGEN WORK CONSTANT
18C	2	CADDS	PTR(31)	ADDR. OF TREEADDS ROUTINE
18C	3	CINSTRLT	FIXED(31)	TREEGEN WORK AREA
190	2	CBASEMIN	FIXED(15)	BASE-BIN MIN ALLOC
194	2	CBINBLK	PTR(31)	ADDR TO BINBLK Q
198	2	CBINSIZE	PTR(31)	RSA BIN SIZE FOR MERGING
19C	2	CBLKITTR	PTR(31)	NEXT UNUSED ITTR
1A0	2	CBUFFER	PTR(31)	ADDR. OF CURRENT WORK BUF
1A4	2	CBLKLSIZ	FIXED(31)	SIZE OF FREE BLOCK LIST
1A8	2	CBUFPTR	PTR(31)	IN - PTR IN BUFF TO VLR
	3	*	PTR(31)	OUT - AVAIL VLR BUFF ADDR
1AC	2	CCHIDX1	FIXED(31)	INDEX AREA 1 HEAD
1B0	2	CCHIDX2	FIXED(31)	INDEX AREA 2 HEAD
1B4	2	CCHIDX3	FIXED(31)	INDEX POOL HEAD
1B8	2	CCHIDX4	FIXED(31)	POOL AREA 1 HEAD
1BC	2	CCHIDX5	FIXED(31)	POOL AREA 2 HEAD
1C0	2	CCON256	FIXED(31)	CONSTANT = 256
1C4	2	CCONNEG4	FIXED(31)	CONSTANT = -4
1C8	2	CCORESAV	FIXED(31)	ORIGINAL SIZE PARM
	2	*	BDY(WORD)	I/O INTERFACE FIELDS
1CC	3	CIOHOLD	FIXED(31)	HOLD SW/Q END
1CC	4	CIOINIT	FIXED(31)	ADDR OF INIT ROUT
1D0	3	CIOSAV13	FIXED(31)	I/O TASK SAVE AREA
1D0	4	CTFCB	FIXED(31)	TFCB CHAIN HEAD
1D4	2	CCURRBLK	PTR(31)	ADDR TO CURR BINBLK
1D8	2	CDCBLIST	PTR(31)	ADDR. OF OPEN/CLOSE LIST
1DC	2	CDATLOC	BDY(WORD)	ICEMAN BASEREGS
1E4	2	CDATLEN	FIXED(31)	DATA BLOCK SIZE
1E8	2	CDECAREA	PTR(31)	ADDR. OF SORTIN/OUT DECB 1EC 2 CDECSIZE
1F0	2	CDELCNT	FIXED(31)	BYTES DELETE IN RSA
1F4	2	CDELETE	PTR(31)	ADDR. OF TREELETE ROUTINE
1F4	3	CBRANCHP	PTR(31)	ADDR. OF TREEGEN BR-TABLE
1F8	2	CDEXCTR	FIXED(31)	HI/LO INDEX COUNTER
1FC	2	CDEXITTR	PTR(31)	INDEX BLOCK ITTR
200	2	CDEXPBUF	FIXED(31)	INDEX ENTRIES PER BUFFER
204	2	CDEXPTRK	FIXED(31)	INDEX ENTRIES PER TRACK
208	2	CDYNAMIC	PTR(31)	ADDR OF UNUSED DYNAMIC AREA
20C	2	CDYNWK	CHAR(8)	DYN ALLOC DEVICE TYPE
20C	3	CDYNWK1	PTR(8)	1ST BYTE OF DYN DEV TYPE
	3	*	CHAR(3)	REST OF CDYNWK 1ST WORD
210	3	CITTRA	FIXED(31)	ALTERNATE ITTR PTR
214	2	CDYNNUMB	FIXED(31)	NUMB OF DYN ALLOC DEVICES
214	3	CITTR	FIXED(31)	ITTR PTR TO 1ST INP STRING
218	2	CDIMSHI	FIXED(16)	HI CONTROL FIELD FOR IMS
21B	2	CDYNLEN	PTR(8)	DYNALLOC FIELD LENGTH
21C	2	CEBUFFRS	FIXED(16)	NUMBER OF E-PHASE BUFFERS
21E	2	CDEBUG	CHAR(1)	DEBUG SWITCH BYTE
21F	2	CENDMODE	CHAR(1)	NO = UNEXPECTED SYSIN EOF
220	2	CEXTRACT	PTR(31)	ADDR. OF EXTRACT ROUTINE
224	2	CEXPAND	FIXED(31)	INT LRECL EXPAND POS OR NEG
228	2	CEMERGE	FIXED(31)	MERGE ORDER - ELIMINATION
22C	2	CEXITPH1	FIXED(31)	LNG OF PH1 RTNS

DISP	LEVEL	LABEL	ATTR	COMMENT
230	2	CEXITPH2	FIXED(31)	LNG OF PH2 RTNS
234	2	CEXITPH3	FIXED(31)	LNG OF PH3 RTNS
238	2	CE18EOD	PTR(31)	USER EOD ADDR
23C	2	CEXIT11	BDY(WORD)	E11 EXIT-
23C	3	CE11NAME	CHAR(8)	NAME
244	3	CE11LNG	FIXED(31)	EXIT LENGTH
244	4	CE11CALL	PTR(31)	CALL ADDRES
248	2	CEXIT15	BDY(WORD)	E15/E32 EXIT -
248	3	CE15NAME	CHAR(8)	NAME
248	4	CE15ADDR	PTR(31)	PASSED ADDRESS
24C	4	CE32MO	FIXED(31)	E32 MERGE ORDER
24D	5	CE32MO24	PTR(24)	
250	3	CE15LNG	FIXED(31)	LENGTH
250	4	CE15CALL	PTR(31)	CALL ADDRESS
254	2	CEXIT16	BDY(WORD)	E16 EXIT
254	3	CE16NAME	CHAR(8)	NAME
25C	3	CE16LNG	FIXED(31)	E16 LENGTH
25C	4	CE16CALL	PTR(31)	CALL ADDR
260	2	CEXIT17	BDY(WORD)	E17 EXIT
260	3	CE17NAME	CHAR(8)	NAME
268	3	CE17LNG	FIXED(31)	E17 LENGTH
268	4	CE17CALL	PTR(31)	CALL ADDRESS
26C	2	CEXIT18	BDY(WORD)	E18 EXIT
26C	3	CE18NAME	CHAR(8)	NAME
274	3	CE18LNG	FIXED(31)	E18 LENGTH
274	4	CE18CALL	PTR(31)	CALL ADDRESS
278	3	CE18XLST	PTR(31)	ADDR TO USER XLIST(5) RTN
27C	3	CE18EXL	PTR(31)	ADDR TO VSAM EXIT LIST
280	3	CE18PSW	PTR(31)	ADDR TO VSAM PASSW. LIST
284	2	CEXIT19	BDY(WORD)	E19 EXIT
284	3	CE19NAME	CHAR(8)	NAME
28C	3	CE19LNG	FIXED(31)	E19 LENGTH
28C	4	CE19CALL	PTR(31)	CALL ADDRESS
290	2	CEXIT61	BDY(WORD)	E61 EXIT
290	3	CE61NAME	CHAR(8)	NAME
298	3	CE61LNG	FIXED(31)	E61 LENGTH
298	4	CE61CALL	PTR(31)	CALL ADDR
29C	2	CEXIT21	BDY(WORD)	E21 EXIT
29C	3	CE21NAME	CHAR(8)	NAME
2A4	3	CE21LNG	FIXED(31)	E21 LENGTH
2A4	4	CE21CALL	PTR(31)	CALL ADDR
2A8	2	CEXIT27	BDY(WORD)	E27 EXIT
2A8	3	CE27NAME	CHAR(8)	NAME
2B0	3	CE27LNG	FIXED(31)	E27 LENGTH
2B0	4	CE27CALL	PTR(31)	CALL ADDR
2B4	2	CEXIT31	BDY(WORD)	E31 EXIT
2B4	3	CE31NAME	CHAR(8)	NAME
2BC	3	CE31LNG	FIXED(31)	E31 LENGTH
2BC	4	CE31CALL	PTR(31)	CALL ADDRESS
2C0	2	CEXIT35	BDY(WORD)	E35 EXIT -
2C0	3	CE35NAME	CHAR(8)	NAME
2C0	4	CE35ADDR	PTR(31)	PASSED ADDRESS
2C8	3	CE35LNG	FIXED(31)	LENGTH
2C8	4	CE35CALL	PTR(31)	CALL ADDRESS
2CC	2	CEXIT37	BDY(WORD)	CE37 EXIT
2CC	3	CE37NAME	CHAR(8)	NAME
2D4	3	CE37LNG	FIXED(31)	E37 LENGTH
2D4	4	CE37CALL	PTR(31)	CALL ADDRESS
2D8	2	CEXIT38	BDY(WORD)	E38 EXIT
2D8	3	CE38NAME	CHAR(8)	NAME
2E0	3	CE38LNG	FIXED(31)	E38 LENGTH
2E0	4	CE38CALL	PTR(31)	CALL ADDRESS
2E4	2	CEXIT39	BDY(WORD)	E39 EXIT
2E4	3	CE39NAME	CHAR(8)	NAME
2EC	3	CE39LNG	FIXED(31)	E39 LENGTH
2EC	4	CE39CALL	PTR(31)	CALL ADDRESS

DISP	LEVEL	LABEL	ATTR	COMMENT
2F0	3	CE39EXL	PTR(31)	ADDR TO VSAM EXIT LIST
2F4	3	CE39PSW	PTR(31)	ADDR TO VSAM PASSW. LIST
2F8	3	CE39XLST	PTR(31)	ADDR TO USER XLIST(5) RTN
2FC	2	CE61PARM		E61 PARM LIST
	3	*	CHAR(3)	ZEROS
2FF	3	CE61PCF	CHAR(1)	CONTR FIELD NUMB
	3	CE61PADD	FIXED(31)	REC ADDRESS
	3	CE61PLEN	FIXED(31)	CONTR FIELD LEN
308	2	CFIELDS	FIXED(31)	NUMBER OF CONTROL FIELDS
30C	2	CFILSIZE	FIXED(31)	USER SPECIFIED FILE SIZE
30D	3	CFILSZ24	PTR(24)	THREE BYTES
310	2	CFILIMIT	PTR(31)	ADDR. OF END OF BUFFER
314	2	CFREECTR	FIXED(31)	AVAILABLE RSA BIN COUNT
318	2	CGMINDEX	FIXED(31)	INDEX IN GETMAIN TABLE
31C	2	CMANMSG	FIXED(31)	PTR TO MSGTABLE FOR RESTORE
320	2	CGENDOPT	BDY(WORD)	SORT GENERATED DEFAULTS:
320	3	COREPARM	FIXED(31)	MAXIMUM ALLOWED
	4	*	CHAR(1)	SIZE OF MAIN STORAGE
321	4	COREPM24	PTR(24)	FOR SORT/MERGE
324	3	CMAXLIM	FIXED(31)	MAX LIMIT FOR SIZE(MAX)
328	3	CMINLIM	FIXED(31)	MIN VIRTUAL STRG FOR PEER
32C	2	CGENDOP1	BDY(WORD)	SORT GEN DEFAULTS CONT:
32C	3	CRESERVE	FIXED(31)	STORAGE
	4	*	CHAR(1)	RESERVED FOR
32D	4	CRSRVD24	PTR(24)	CALLING PROGRAM
330	3	CMSGOPTN	CHAR(4)	MESSAGE OPTION
330	4	CCONSMOD	CHAR(1)	Y = MESSAGES TO CONSOLE
331	4	CPRTRMOD	CHAR(1)	Y = MESSAGES TO PRINTER
332	4	CPRINMOD	CHAR(1)	Y = PRINT ALL MESSAGES
333	4	CCRITMOD	CHAR(1)	Y = PRINT CRITICAL MSG
334	3	CEQUALS	CHAR(1)	Y = PRESERV ORDER OF EQUALS
335	3	CLISTMOD	CHAR(1)	Y = LIST CONTROL CARDS
336	3	CERRU016	CHAR(1)	Y = CRITICAL ERROR ABEND
337	3	CERRINV	CHAR(1)	Y = CRITICAL ERROR ABEND
338	3	CSYSNAME	CHAR(8)	NAME FOR PRINT DD CARD
340	3	CVIO	CHAR(1)	VIO UNDER MVS
341	3	CEXMODE	CHAR(1)	R=EXCPVR V=EXCP X=MVT
342	3	CRESDNT	CHAR(1)	A=ALL M=MAN N=NONE
343	3	CRELSE	CHAR(1)	Y=RELEASE WORKSPACE
344	3	CSECALC	CHAR(1)	Y=AUTOM. SEC. ALLOC.
345	3	CVERIFY	CHAR(1)	Y=VERIFY OUTPUT RECS
346	3	CBLKSET	CHAR(1)	Y=BLOCKSET MAY BE USED
347	3	CCHALT	CHAR(1)	C=TRANSLATE CH AND AQ A=TRANSLATE AQ ONLY
348	3	CSVC	PTR(16)	SVC INSTRUCTION
34A	3	CCHECK	CHAR(1)	Y=CHECK COUNTERS N=COUNTER CHECK SUPPRESS
34B	3	CSMF	CHAR(1)	S=SHORT SMF RECORD F=FULL SMF RECORD
34C	3	CVBLKSET	CHAR(1)	N =BYPASS VLR-BLOCKSET Y =VLR-BLOCKSET MAY BE USED
34D	2	CSTAE	CHAR(1)	Y = STAE WANTED AT ABEND
34E	2	CVIOREC	CHAR(1)	VIO OR TAPE REC
	2	*	CHAR(2)	RESERVED
352	2	CIDXPLB	FIXED(15)	BLOCKS/LOGIC INDEX
354	2	CGETMTMP	PTR(31)	ADDR. OF TEMP. GETM. AREAS
354	3	CGETTMP	PTR(31)	ALTERNATE NAME
358	2	CGETREC	PTR(31)	ADDR. OF INPUT GET ROUTINE
358	3	CTIOTIN	PTR(31)	PTR TO INPUT DD NAME (DEF)
35C	2	CGETPRM	PTR(31)	PERM AREAS CHAIN
360	2	CHECKDCB	FIXED(31)	CHECKPOINT DCB POINTER
364	2	CHKTTSTR	PTR(31)	CHKPT ITT SAVE AREA
368	2	CHBUF1	FIXED(31)	INDEX HDR BFR ADDR
36C	2	CHKBLK	FIXED(31)	TOTAL BLKS EST FOR PRIM EXT
370	2	CINSIZE	FIXED(31)	SORTIN LOGICAL RCD LENGTH
374	2	CIBUFFER	PTR(31)	ADDR. OF INPUT INDEX BUFFER
378	2	CIBSPACE	FIXED(31)	INPUT BFR SPACE
37C	2	CINDEX	PTR(31)	ADDR. OF INPUT INDEX ENTRY

DISP	LEVEL	LABEL	ATTR	COMMENT
380	2	CINDEXHI	PTR(31)	ADDR. OF END OF INDEX BUF
384	2	CIDEXSIZ	FIXED(31)	INDEX ENTRY SIZE
388	2	CITTFQ	PTR(31)	HEAD OF FREE TRACK HOLDERS
388	3	CSAVREG7	FIXED(31)	SAVE INPUT PTR (ICELIM)
38C	2	CITTWQ	PTR(31)	HEAD OF AVAIL. TRACK QUEUE
38C	3	CONBLKSZ	FIXED(15)	BLKSIZE OF 1-ST SORTIN FILE
390	2	CIDXLEN	FIXED(31)	INDEX BLOCK SIZE
394	2	CINLOC	FIXED(31)	ADDR TO INPUT REC
398	2	CKEYSIZE	FIXED(31)	SIZE OF CONTROL WORD
39C	2	CLOCSIZE	BDY(WORD)	TABLE FOR VARIABLE GETMAIN
39C	3	COMLOC	PTR(31)	GETMAINED CHAIN
3A0	3	COMSIZE	FIXED(31)	SIZE OF RETURNED STORAGE
3A4	2	CBASESIZ	FIXED(31)	BASE-BIN SIZE @PTM411M
3A8	2	CLASTIME	FIXED(31)	LAST ENTRY TO REAL TIME RTN
3AC	2	CLOCKS		TIME ACCUMULATORS
3AC	3	CLOCK0	FIXED(31)	C-PHASE ELAPSED TIME
3B0	3	CLOCK1	FIXED(31)	C-PHASE CPU TIME
3B4	3	CLOCK2	FIXED(31)	P-PHASE ELAPSED TIME
3B8	3	CLOCK3	FIXED(31)	P-PHASE CPU TIME
3BC	3	CLOCK4	FIXED(31)	R-PHASE ELAPSED TIME
3C0	3	CLOCK5	FIXED(31)	R-PHASE CPU TIME
3C4	3	CLOCK6	FIXED(31)	E-PHASE ELAPSED TIME
3C8	3	CLOCK7	FIXED(31)	E-PHASE CPU TIME
3CC	3	CLOCK8	FIXED(31)	SPECIAL CLOCK 0
3D0	3	CLOCK9	FIXED(31)	SPECIAL CLOCK 1
3D4	3	CLOCK10	FIXED(31)	SPECIAL CLOCK 2
3D8	3	CLOCK11	FIXED(31)	SPECIAL CLOCK 3
3DC	3	CLOCK12	FIXED(31)	SPECIAL CLOCK 4
3E0	3	CLOCK13	FIXED(31)	SPECIAL CLOCK 5
3E4	2	CLCNUM	FIXED(31)	NO. OF CLC-INSTR. TREEGEN
3E4	3	CBLKIN	FIXED(15)	INPUT BLOCKSIZE
3E6	3	CLTHIN	FIXED(15)	INPUT RECORD LENGTH
3E8	2	CLCLTH	FIXED(31)	LENGTH LAST CLC. TREEGEN
3E8	3	CRECFMIN	CHAR(1)	INPUT RECORD FORMAT
	3	*	CHAR(3)	NOT USED AFTER PH 0
3EC	2	CMOVEKEY(3)	PTR(16)	CTL WORD MVC OR BAL TO RTN
3F4	2	CMERGE	PTR(31)	ADDR. OF TRMERGE ROUTINE
3F4	3	CADDTAB	FIXED(31)	TREEGEN WORK AREA
3F8	2	CMSGTYPE	PTR(31)	DEBUG MESSAGE TYPE INDEX
3FC	2	CMORETRK	PTR(31)	TRKS NOT TO RELEASE
400	2	CMINMAX	BDY(WORD)	PARMS FOR VARIABLE GETMAIN
400	3	CMINSIZE	FIXED(31)	MINIMUM SIZE ACCEPTED
404	3	CMAXSIZE	FIXED(31)	MAXIMUM SIZE WANTED
408	2	CMAINSIZ	FIXED(31)	SIZE OF WORKING STORAGE
40C	2	CMINREC	FIXED(31)	MINIMAL RECORD SIZE
40E	3	CMINRECH	FIXED(15)	HALF-WORD
410	2	CMODEVAL	FIXED(31)	MODAL RECORD SIZE
412	3	CMODVALH	FIXED(15)	HALF-WORD
414	2	CMODSTM	BDY(WORD)	MODS STATEMENT -
414	3	CMODSTRT	PTR(31)	STARTING BYTE
418	3	CMODSEND	PTR(31)	ENDING BYTE
41C	2	CMODULE	CHAR(8)	MOD TO BE CALLED
41C	3	CLINK	FIXED(31)	ZERO IF NO MORE MOD TO CALL
41F	4	CMODNAME	CHAR(3)	SIGNIFICANT PART OF MODNAME
424	2	CMOVERB(3)	PTR(16)	RCD MVC RSA TO BUFFER
42A	2	CMOVERBC(3)	PTR(16)	RCD MVC RSA TO BUFFER(C-PH)
430	2	CMOVEBR(3)	PTR(16)	CCD MVC BUFFER TO RSA
436	2	CMOVEBRI(3)	PTR(16)	INDEX MVC BUFFER TO RSA
43C	2	CMOVERBI(3)	PTR(16)	INDEX MVC RSA TO BUFFER
442	2	CMOVEE35(3)	PTR(16)	MVC FOR USER RTN AT E35
448	2	CMODSLIB	CHAR(8)	MODS LIBRARY NAME
450	2	CMSGTABL	PTR(31)	ADDR. ERROR MESSAGE TABLE
454	2	CMOVELNG	FIXED(31)	BYTES TO MOVE TO RSA
458	2	CNUMWORK	FIXED(31)	NUMBER OF WORK AREAS
45C	2	CNEXTWRK	FIXED(31)	PRIM WORKD CHAIN

```
DISP LEVEL      LABEL        ATTR          COMMENT
460   2         COBSPACE     FIXED(31)     OUTPUT BFR SPACE
464   2         COPSRTIN     PTR(31)       OPEN PTR USED IN DEF + CRE
468   2         COUTSIZE     FIXED(31)     SORTOUT LOGICAL RCD LENGTH
46C   2         COUTLIM      PTR(31)       ADDR. OF END OF WORK BUF
470   2         COUNTERS                   EVENT COUNTERS
470    3        COUNTER0     FIXED(31)     ENTRIES TO  INOUTRTN
474    3        COUNTER1     FIXED(31)     ISSUED EXCPS
478    3        COUNTER2     FIXED(31)     RECORDS IN
47C    3        COUNTER3     FIXED(31)     RECORDS OUT
480    3        COUNTER4     FIXED(31)     RECORDS INSERTED
484    3        COUNTER5     FIXED(31)     RECORDS DELETED
488    3        COUNTER6     FIXED(31)     SPECIAL COUNT 0
48C    3        COUNTER7     FIXED(31)     SPECIAL COUNT 1
490    3        COUNTERX     FIXED(31)     TEMPORARY COUNT 1
494   2         CONSW        PTR(8)        MISCELLANEOUS SWITCHES
494    3        CSKIP        BIT(1)        SKIP PHASES CONTROL SWITCH
494    3        CSTOSW       BIT(1)        TO-REG IN RCD-MOVE SAVED
494    3        CSFROMSW     BIT(1)        FROM-REG IN RCD-MOVE SAVED
494    3        CTECHNQS     BIT(1)        TECHNIQUE FORCED
494    3        CDDFIRST     BIT(1)        PROC.FIRST DDNAME CHARACTER
494    3        CERRSW1      BIT(1)        ERROR IN REQUIRED PARAMETER
494    3        CERRSW2      BIT(1)        ERROR IN OPTIONAL PARAMETER
494    3        CNEWPAGE     BIT(1)        NEW PAGE HAS BEEN TAKEN
495   2         CONSW1       PTR(8)        MISCELLANEOUS SWITCHES
495    3        CCONCASW     BIT(1)        CONCATENATION OF DATASETS
495    3        CFIRSTSW     BIT(1)        OPEN OF FIRST DATA SET
495    3        CSHORTSW     BIT(1)        SHORT VAR RECORD FOUND
495    3        CDYNSW       BIT(1)        ONE DYNALLOC FAILED
495    3        CTPDYNSW     BIT(1)        NO DYNNUMB FOR TAPE
495    3        CALLOCND     BIT(1)        ALL WORKFILES ALLOCATED
495    3        CFIXPAGE     BIT(1)        PROTECTED AREA FIXED
495    3        CPARMODE     BIT(1)        PARTITION MODE SWITCH
495     4       CIMS         BIT(1)        SORT INVOKED BY IMS
496   2         CONSW2       PTR(8)        MISCELLANEOUS SWITCHES
496     4       CVSAMR1      BIT(1)        VSAM RELEASE 1 IN SYSTEM
496     4       CNOREUSE     BIT(1)        VSAM OUTPUT NON-REUSABLE
496     4       CDYNGEN      BIT(1)        DYNALLOC GENERIC NAMES USED
496     4       CINCHAIN     BIT(1)        NO CHAIN SCHED FOR SORTIN
496     4       CONCHAIN     BIT(1)        NO CHAIN SCHED FOR SORTOUT
496     4       CDKALLOC     BIT(1)        SORTDK ALLOCATED
496     4       COPENCLS     BIT(1)        OP/CL SYSOUT IN PRSS
496     4       CFAKE35      BIT(1)        L2 GET L3 SWITCH
497   2         CONRECFM     CHAR(1)
498   2         COPTIONS                   OPTION SWITCHES
498    3        CSORTNAM     CHAR(4)       PREFIX FOR DD STATEMENTS
49C    3        CSRTMRGE     CHAR(1)       SORT=(S) |  MERGE=(M)
49D    3        CSYSDUMP     CHAR(1)       YES = SYSUDUMP OR SYSABEND
49E    3        CDIAGMOD     CHAR(1)       YES = PRINT DIAGNOSTICS
49F    3        CINVOKED     CHAR(1)       YES = DYNAMIC INVOCATION
4A0    3        CCHECKPT     CHAR(1)       YES = TAKE CHECKPOINTS
4A2    3        CSEEKMOD     CHAR(1)       YES = DO STAND-ALONE SEEKS
4A3    3        CCONMODE     CHAR(1)       Y=PEER  N=NONPEER  0=5734
4A4    3        CE15MOD      CHAR(1)       YES = E15 ADDRESS GIVEN
4A5    3        CE35MOD      CHAR(1)       YES = E35 ADDRESS GIVEN
4A6    3        CMODCARD     CHAR(1)       YES = MODS STMNT PRESENT
4A7    3        CEXACTSZ     CHAR(1)       YES = EXACT  FILSZ  GIVEN
4A8    3        CESTIMSZ     CHAR(1)       YES = ESTIM.  SIZE|FILSZ
4A9    3        CTECHNQ      CHAR(1)       P|B|O|X|N  OR DEFAULT  E
4AA    3        CKEYMODE     PTR(8)        NUMBER OF INDEXES PER TRACK
4AB    3        CSYSTEM      CHAR(1)       0=MFT/MVT  S=SVS  M=MVS
4AC    3        CPAGESIZ     PTR(31)       PAGE SIZE FOR SYSTEM
4B0   2         COMMB        PTR(31)       EXTRACT/RESTORE CODE AREA
4B4   2         COMMBEND     PTR(31)       END OF EXTRACT/RESTORE AREA
4B8   2         CONLRECL     FIXED(15)
4BA   2         CPOSIZE      FIXED(15)     PARTITION ZERO SIZE
```

DISP	LEVEL	LABEL	ATTR	COMMENT
4BC	2	CPRTDECB(5)	PTR(31)	PRINTER OUTPUT DECB
4D0	2	CPAGELIM	PTR(31)	END PTR TO UNPROT FIXD AREA
4D4	2	CPAGELOC	PTR(31)	ADDR. OF FIXED AREAS TABLE
4D8	2	CPEERAGE	FIXED(31)	MIN. NO STRINGS ON SORTWK
4DC	2	CPARCLC(3)	PTR(16)	PAR INDEX BIN COMP
4E2	2	CPARMVC(3)	PTR(16)	PAR INDEX BIN MOVE
	2	*	CHAR(2)	RESERVED **********
4EA	2	CPRT	CHAR(129)	(WORD 3) PRINT BUFFER
4EA	3	CPRTHEAD(4)	FIXED(15)	VLR AND VLB INFORMATION
4F2	3	CPRTLINE	CHAR(121)	(WORD 3) PRINTER OUTPUT
	4	*	FIXED(15)	PRINT LINE
4F4	4	CMSGLIST(29)	FIXED(31)	MSG VARIABLE INFO LIST
	4	*	CHAR(3)	
56B	2	CPREGION	CHAR(1)	REGION IS CORE LIMIT
56C	2	CPHASECG	FIXED(31)	MAXIMUM G-VALUE - CREATION
570	2	CPHASEPG	FIXED(31)	MAXIMUM G-VALUE - PARTTION
574	2	CPHASERG	FIXED(31)	MAXIMUM G-VALUE - REDUCTION
578	2	CPHASEEG	FIXED(31)	MAXIMUM G-VALUE-ELIMINATION
57C	2	CPHASEWK	BDY(WORD)	PHASE WORK AREA
57C	3	CEXITTAB	CHAR(16)	EXIT RETURN BRANCHTAB
57C	4	CDECL1	FIXED(31)	REC LEN1 IN DEC
580	4	CDECL2	FIXED(31)	REC LEN2 IN DEC
584	4	CDECL3	FIXED(31)	REC LEN3 IN DEC
588	4	COBUFFER	PTR(31)	ADDR OF OUTPUT INDEX BFR
588	5	CDECDED	PTR(31)	ICEDEC - ICEDED COMM
588	6	CDEVWK	PTR(31)	PTR TO DEV WORK AREA
58C	3	CE15PARM	CHAR(4)	E15 PARAMEYER LIST
590	3	CE35PARM	CHAR(12)	E35 PARAMETER LIST
59C	2	CPUTREC	PTR(31)	ADDR. OF OUTPUT PUT ROUTINE
59C	3	CTIOTOUT	PTR(31)	PTR OUTPT DD NAME (DEF-DEG)
5A0	3	CORMSG39	PTR(31)	CORE VALUE FOR MSG 39 (DEV)
5A0	2	CPAGERSA	PTR(31)	PTR TO END OF LAST BUFFD0
5A4	2	CPBUF1	FIXED(31)	INDEX POOL ADDR
5A8	2	CPROTSIZ	FIXED(31)	PROT AREA SIZE
5AC	2	CPRMSIZ	FIXED(31)	PERM ALLOC SIZE
5B0	2	CREBUILD	PTR(31)	REBUILD SPANNED RECORDS
5B0	3	COREUSED	FIXED(31)	CORE USED FOR MSG 92/93
5B4	2	CRECSIZE	FIXED(31)	SIZE OF TRANSPOSED RECORD
5B8	2	CRECADDR	PTR(31)	ADDR. OF RECORD - FOR VSAM
5B8	3	CSPNVSAM	PTR(31)	
5BC	2	CRECAREA	PTR(31)	VLR PAD/RESTORE AREA
5C0	2	CRECCTR	FIXED(31)	RCD COUNT FOR CURRENT TRACK
5C4	2	CRECORD	FIXED(31)	ADDR TO RECORD KEY
5C8	2	CRECPTR	PTR(31)	SPANNED REC PTR
5CC	2	CRECPBUF	FIXED(31)	RECORDS/BUFFER
5D0	2	CRECTBUF	FIXED(31)	RCD/BUF *TREE ENTRY SIZE(4)
5D4	2	CRBUFFRS	FIXED(16)	NUMBER OF R-PHASE BUFFERS
5D6	2	CRETCODE	PTR(8)	RETURN CODE AREA (0|10)
5D7	2	CRECTYPE	PTR(8)	RECORD TYPE
5D7	3	CFIXREC	BIT(1)	ON = FIXED LENGTH RECORDS
5D7	3	CVARREC	BIT(1)	ON = VARIABLE LENGTH RECS
5D7	3	CSPANIN	BIT(1)	ON = VAR. SPANNED INPUT
5D8	2	CRESPMTM	PTR(31)	PERM RES CH OR TEMP RES
5DC	2	CRESTORE	PTR(31)	ADDR. OF RESTORE ROUTINE
5E0	2	CRESTMP	PTR(31)	TEMP RESERVE CNT
5E4	2	CRLSE	CHAR(1)	YES FOR RELEASE OF SORTWK
5E5	2	CONSW3	PTR(8)	MISCELLANEOUS SWITCHES
5E5	3	CB37MSG	BIT(1)	B37 RECOVERY MAG FLAG @L5A
5E5	3	CWKVAMDS	BIT(1)	VIO WORK DATASET PRESENT
5E5	3	CSRTCNTL	BIT(1)	SORTCNTL STATEMENT PRESENT
5E5	3	CABRETRN	BIT(1)	TO PREV SEC. ENTR OF ABENDR
5E5	3	CMERGEIN	BIT(1)	SORTIN01 DD STMNT PRESENT
5E8	2	CRMERGE	FIXED(31)	MERGE ORDER - REDUCTION
5EC	2	CRPLIN	PTR(31)	ADDR OF VSAM INPUT RPL
5F0	2	CRPLOUT	PTR(31)	ADDR OF VSAM OUTPUT RPL
5F4	2	CRSAFREE	PTR(31)	ADDR. OF NEXT BIN IN RSA
5F4	3	CE35AREA	PTR(31)	

DISP	LEVEL	LABEL	ATTR	COMMENT
5F8	2	CSYSIN	PTR(31)	SYSIN DCB POINTER
5FC	2	CSYSOUT	PTR(31)	SYSOUT DCB POINTER
600	2	CSORTIN	FIXED(31)	SORTIN DCB POINTER
604	2	CSORTOUT	FIXED(31)	SORTOUT DCB POINTER
608	2	CSKIPREC	FIXED(31)	NUMBER OF RECORDS TO SKIP
609	3	CSKIPR24	PTR(24)	THREE BYTES
60C	2	CSCANLOC	PTR(31)	ADDR. OF LAST SCAN LOCATION
60C	3	CMAXREC	FIXED(31)	MAXIMUM RECORD SIZE
610	2	CSAVITTR	PTR(31)	PREVIOUS INDEX BLOCK ITTR
614	2	CSTRINGS	FIXED(31)	STRING COUNT
618	2	CSELECT	PTR(31)	ADDR. OF TSELECT ROUTINE
61C	2	CSORTWK	PTR(31)	PTR TO SORTWK DCB LIST
620	2	CSYSOUTD	PTR(31)	SYSOUT - DCB
680	2	CSUBSIZE	FIXED(15)	SUB-BIN SIZE
682	2	CSUBPBAS	FIXED(15)	SUB-BIN/BASE-BIN
684	2	CSECALL	CHAR(2)	SECONDARY ALLOCATION SWITCH
688	2	CSTAESAV	PTR(31)	STAE WORK AREA -
	3	*	PTR(8)	SAVE ADDRESS -
689	3	CSTAES24	PTR(24)	FOR NEXT LEVEL
68C	2	CSECTRK	FIXED(31)	SEC.ALLOC. TRK COUNTER
690	2	CSAVEPTR	FIXED(31)	SAVE AREA POINTER
694	2	CSYSRES	FIXED(31)	SYSTEM NEEDS
698	2	CTREESIZ	FIXED(31)	SIZE OF ALLOCATED TREE
69C	2	CTREELOC	PTR(31)	ADDR. OF START OF TREE
6A0	2	CUTABENT	PTR(31)	PTR/DISPL TO UNIT TAB ENTRY
6A0	3	CBLKPTRK	FIXED(31)	BLKS/TRK LOACALLY
6A4	2	CTRESHLD	FIXED(31)	FREE BYTES IN RSA
6A8	2	CTMPSIZ	FIXED(31)	TEMP ALLOC SIZE
6AC	2	CTREE2	PTR(31)	LOOK AHEAD TREE
6B0	2	CTESTKEY(3)	PTR(16)	CLC FOR TWO CONTROL WORDS
6B8	2	CURRDECB	PTR(31)	ADDR. OF CURRENT DECB
6BC	2	CURRDATE	FIXED(31)	CURRENT DATE - PACKED DEC.
6C0	2	CURREC	PTR(31)	ADDR TO BUFFERED REC IN RSA
6C4	2	CVERACC	FIXED(31)	ACCUMULATOR FOR VERIFY
6C8	2	CVSAMERR	PTR(31)	ADDR OF MODULE ICEERR
6CC	2	CVSAMRVE	FIXED(31)	STORAGE RESEVED FOR VSAM
6D0	2	CVMOVEL	FIXED(31)	VLR - NO OF BYTES TO MOVE
6D4	2	CVREMSW	BIT(8)	ICEVRE MISC SWITCH
6D4	3	CINCORE	BIT(1)	1=INCORE SORT
6D4	3	CVIM	BIT(1)	1=BYPASS VED CALL VIM
6D4	3	CE15DELT	BIT(1)	1=E15 DELETED
6D4	3	CRSAFULL	BIT(1)	1=RSA IS FULL
6D4	3	CFRSTBIN	BIT(1)	1=FIRST BIN ALLOC
	3	CFIRSTSB	BIT(1)	1=FIRST SUB-BIN
6D5	2	CWRKTYPE	CHAR(1)	T=TAPE D=DISK
6D6	2	CWMINCON	FIXED(15)	CTRL FIELD BASED MIN RECLEN
6D8	2	CWTO	BDY(WORD)	WTO - MF=L
	3	*	FIXED(15)	TEXT LENGTH
	3	*	PTR(16)	MCC FLAGS
6DC	3	CMESSAGE	CHAR(120)	MESSAGE AREA
	4	*	CHAR(3)	MESSAGE IDENTIFICATION
6DF	4	CMSGNO	CHAR(3)	MESSAGE NUMBER
6E2	4	CMSGTAG	CHAR(1)	MESSAGE TAG
	4	*	CHAR(1)	MESSAGE BLANK
6E4	4	CJOBNAME	CHAR(8)	JOB NAME
6EC	4	CDOT	CHAR(1)	DOT
6ED	4	CSTEPNAM	CHAR(8)	STEP NAME
6F5	4	CMSGBLNK	CHAR(1)	MESSAGE BLANK
6F6	4	CMSGINFO	CHAR(94)	MESSAGE TEXT
	3	*	PTR(16)	DESCRIPTOR CODES
	3	*	PTR(16)	ROUTING CODES
758	2	CWORKQ	PTR(31)	ADDRESS OF WORK AREA QUEUE
75C	2	CWBPOOL	FIXED(31)	WB-POOL ADDRESS
760	2	CWBPOOLL	FIXED(15)	WRITEBACK POOL SIZE
762	2	CWBPOOLC	FIXED(15)	CURRENT POOL SIZE

DISP	LEVEL	LABEL	ATTR	COMMENT
764	2	CWBPOOLR	FIXED(15)	OFFSET TO RESERVE BLOCK
766	2	CWMINREC	FIXED(15)	MIN EXTRACT IN NON-PEER
768	2	CEXIT25	BDY(WORD)	EXIT25
768	3	CE25NAME	CHAR(8)	EXIT E25 NAME
770	3	CE25LNG	FIXED(31)	EXIT E25 LENGTH
770	4	CE25CALL	PTR(31)	AND ITS CALL ADDRESS
774	2	CSRTMODS	PTR(31)	SORTMODS DCB POINTER
774	3	CSYSLIN	PTR(31)	SYSLIN DCB POINTER
778	2	CLINKREQ	CHAR(1)	FLAGS LINKEDITING REQUEST
779	2	CLODCNT1	CHAR(1)	LOAD COUNT PHASE 1
77A	2	CLODCNT2	CHAR(1)	LOAD COUNT PHASE 2
77B	2	CLODCNT3	CHAR(1)	LOAD COUNT PHASE 3
77C	2	CINDEX1	PTR(31)	BUFFER POINTERS
780	2	CINDEXH1	PTR(31)	
784	2	CDECAREB	PTR(31)	USED IN ICELINK
788	2	CTABPTR	PTR(31)	POINTER IN SCANTAB
78C	2	CEXITLOA	PTR(31)	EXIT LOAD ROUTINE ADDRESS
790	2	CEXITDEL	PTR(31)	EXIT DELETE ROUTINE ADDR.
794	2	CEQE25	PTR(31)	PICK UP EQUALS FIELD
794	3	CEQE25OP	CHAR(2)	OP CODE ICM
798	2	CSTWNAME	CHAR(12)	FOR STOW MACRO
7A4	2	COUNTERY	FIXED(31)	RECS DELETED BY E25
7A8	2	CLISTMAD	CHAR(1)	TEMP SAVE FOR CLISTMOD
7A9	2	CPHASLNK	CHAR(1)	TO REMEMBER THAT WE LINK
7AA	2	CEXCPIN	CHAR(1)	EXCP CAN BE USED FOR SORTIN
7AB	2	CEXCPOUT	CHAR(1)	EXCP -//- SORTOUT
7AC	2	CBUGADDR	FIXED(31)	ICEBUG AREA ADDRESSES
7B0	2	CMODDCB	FIXED(31)	MODSLIB DCB ADDRESS
7B4	2	CSYSLMOD	FIXED(31)	SYSLMOD DCB ADDRESS
7B8	2	CZONE61	CHAR(32)	INTERM. STOR. F. ZD W. E61
7D8	2	COBSEXIN	PTR(31)	ADDRESS OF SORTIN OPEN EXIT
7DC	2	COBSEXUT	PTR(31)	ADDRESS OF SORTOUT -//-
7E0	2	CSTOPIN	CHAR(1)	CANT USE EXCP FOR SORTIN
7E1	2	CSTOPOUT	CHAR(1)	CANT USE EXCP FOR SORTOUT
7E2	2	CDISPMOD	CHAR(1)	DISP IS MOD FOR SORTOUT
7E3	2	CMULTVOL	CHAR(1)	MULTIVOLUME DATA SET FLAG
7E4	2	COBSEXIT	PTR(31)	EXCP OPEN EXIT ADDRESS
7E8	2	COBSBLOK	PTR(31)	ADDRESS OF EXCP DATA CODE
7EC	2	CBLDLNAM	CHAR(8)	MEMBER NAME FOR SORTIN
7F4	2	CSAVER14	FIXED(31)	SAVE AREA IN ALL PHASES
7F8	2	CINCONCA	CHAR(1)	CONCATENATED SORTIN FLAG
7F9	2	COPENFL	CHAR(1)	PERFORM OPEN IN ICEOBS
7FA	2	CMSGACNT	FIXED(15)	COUNT OF ACTION MESSAGES
7FC	2	CLENSTAT	PTR(31)	SMF WORK AREA PTR
800	2	CXTRSTAT	PTR(31)	SMF STATISTICS RTN PTR OR EXTRACT RTN PTR
804	2	CFILBYTE	FIXED(31)	FILE SIZE ACCUM (BYTES)
808	2	CTOTTIME	FIXED(31)	CPU TIME AREA FOR SMF
80C	2	CAVGRCDL	FIXED(31)	AVGERAGE RECORD LENGTH
810	2	CTREEBAS	FIXED(31)	TREE BASECODE ADDRESS
	2	CPATCH	FIXED(31)	RESERVED
828	2	CTRACE	CHAR(128)	TRACE TABLE

This page intentionally left blank.

Control Phase Information Area (CPI), Conventional Techniques

The program's Control Phase Information Area is set up by ICERC0 and ICERCM. It is used for the communication of parameters that are needed by the general assignment routines. It is expanded into the PPI (Phase-to-Phase Information Area) by ICERC1 in the Optimization Phase.

DISPLACEMENT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION MEANING/USE
0	(0)	72	ICERC5	Register area
72	(48)	72		Control Field Information
72	(48)	2	CPINUMCF	No of control fields
74	(4A)	72	CPIPCFxx	12 entries in the form:
		3		- Position
		2		- Length
		1		- Format + sequence
146	(92)	8	CPISW1	Switch of 64 Bits (See PPISW1 for details)
154	(9A)	4	CPILAB09	Output buffers:
			(Byte 1)	No of ph2 output buffers
			(Byte 2)	No of ph3 output buffers
			(Bytes 3-4)	Size of ph3 output buffers
158	(9E)	2	CPINWKU	No of work units
160	(A0)	2	CPIP1ASZ	Ph1 assignment size
162	(A2)	2	CPIIPBLK	Input blocking
164	(A4)	2	CPIFFF	Displacement of F field (used by extract routine)
166	(A6)	2	CPIPBUFF	Displacement of packing buffer
168	(A8)	2	CPIP2ASZ	Size of ph2 assignment modules
170	(AA)	2	CPIP3ASZ	Size of ph3 assignment modules
172	(AC)	2	CPIOPBLK	Output blocking
174	(AE)	2	CPIBUF23	No of buffers (ph2,3)
176	(B0)	2	CPISRTBL	B (blocking used on work areas)
178	(B2)	2	CPIRCDL1	l_1 from RECORD stmt (LENGTH)
180	(B4)	2	CPIRCDL2	l_2 from RECORD stmt (LENGTH)
182	(B6)	2	CPIRCDL3	l_3 from RECORD stmt (LENGTH)
184	(B8)	2	CPIRCDL4	l_4 from RECORD stmt (LENGTH)
186	(BA)	2	CPIRCDL5	l_5 from RECORD stmt (LENGTH)
188	(BC)	2	CPILAB03	Size of ph1 input buffers
190	(BE)	2	CPILAB07	Output buffer information
192	(C0)	2	CPIPGCSZ	Size of area used by extract rtn
194	(C2)	2	CPIBINSZ	Bin size
196	(C4)	3	CPISKPRD	Count of skipped records
199	(C7)	3	CPIATP1E	Addr of invoking pgm's ph1 exit
202	(CA)	3	CPIMODEX	Exits Activated (see first 3 bytes of PPIMODEX for details)
205	(CD)	3	CPIATP3E	Addr of invoking pgm's ph3 exit
208	(D0)	3	CPITAVLC	Size of available core, ph1
211	(D3)	3	CPIFILSZ	Value of FILSZ/SIZE parameter
214	(D6)	3	CPISRTG	G (Capacity of RSA, in records)
217	(D9)	3	CPISIRG	Size of interblock gap (tape wk)
220	(DC)	3	CPIP1GC	Size of ph1 work area
220	(DC)	3	CPIP1RSZ	Ph1 running modules size
223	(DF)	3	CPIP2GC	Size of ph2 work area
223	(DF)	3	CPIP2RSZ	Ph2 running modules size
226	(E2)	3	CPIP3GC	Size of ph3 work area
226	(E2)	3	CPIP3RSZ	Ph3 running modules size
229	(E5)	1	CPIBUF1	No of ph1 buffers
230	(E6)	1	CPIMRGMX	Maximum merge order

DISPLACEMENT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION MEANING/USE
231	(E7)	1	CPIMRGAL	Alternate merge order
232	(E8)	3	CPILINK	Link Edit Info for exit rtns
		xxxx xxxx	(Byte 1)	Exits in order as for PPIMODEX:
		xxxx xxxx	(Byte 2)	1=Routine was link edited,
		x...	(Byte 3)	0=Not edited (via the program)
		.1..		E11 routine edited separately
		..1.		E21 routine edited separately
		...1		E31 routine edited separately
		.... 1...		Linkage editing was carried out
		.... .1..		VSAM output not reusable
		.... ..1.		Linkage edit error
		1		VSAM release 1 in system
235	(EB)	3	CPIADDCF	When more than 12 ctl fields: addr of additional information
238	(EE)	8	CPISORCE	DDname of user mod library
246	(F6)	4	CPIDDSRT	4 chars to replace SORT in DDnames (fr param list when pgm invoked)
250	(FA)	2	CPIDCBIN	Size of SORTIN DCB
252	(FC)	2	CPIDCBOU	Size of SORTOUT DCB
254	(FE)	4	CPIXCAP	OSCL:RMAX (max no of bytes/reel) Otherwise: Capacity of interm. storage (in records)
258	(102)	16	CPIDEV	Device inf: words 2-5 fr DEVTYPE macro (used by ICERCI)
274	(112)	3	CPIAUENT	Addr of user exit name table
277	(115)	1	CPISUENT	Size of user exit name table
278	(116)	8	CPIMSGDD	Message DDname if not SYSOUT
286	(11E)	1	CPIRECFM	User record format for ph3
287	(11F)	8	CPIJOBNM	Jobname
295	(127)	8	CPISTPNM	Stepname
303	(12F)	1	CPISW2	Switch of 8 Bits
		1...	CPIFSEON	FILSZ given with exact value
		.1..		Exec sort with ABEND option
		..1.		AQ control fields present
		...1		Preserve order of equals
		.... 1...		VSAM input
		.... .1..		KSDS VSAM output
		.... ..1.		ESDS VSAM output
		1		SIZE(MAX) used
306	(130)	2	CPIMXCOL	Rightmost col included in any control field
308	(132)	1	CPIDEVCD	Device code for unit with RPS
309	(133)	256	CPIAQTAB	Translate table for use with AQ control fields

Phase-to-Phase Information Area (PPI), Conventional Techniques

The Phase-to-Phase information area (PPI) is a communication area for all modules of the conventional sort/merge program. It is created during Phase 0 and resides in main storage throughout execution. It is not executable, and has the CSECT name ICERCA.

The program uses register 13 as the base register for the PPI. Reference to any field within the area can be made by adding the appropriate displacement to the contents of register 13. The displacement for each field within the PPI appears in decimal and hexadecimal form in the description which follows Figure 15.

Abbreviations

Mult	means a multiple compare routine is used to compare records;
No mult	means it is not;
Extract	means an extract routine is used to compare records.
Fix	means records are fixed length;
Var	means records are variable length.

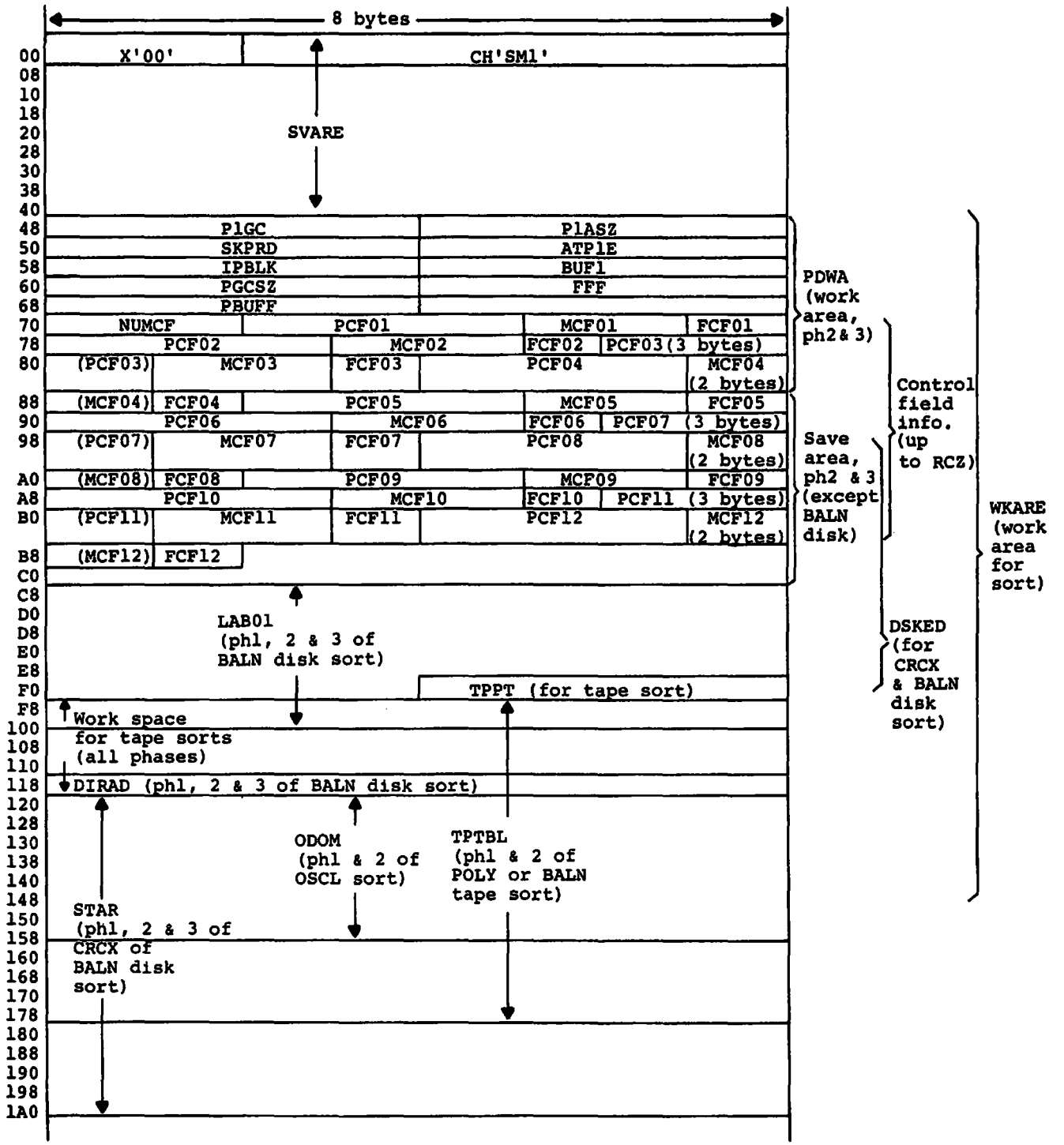


Figure 15. (1 of 3) Map of PPI

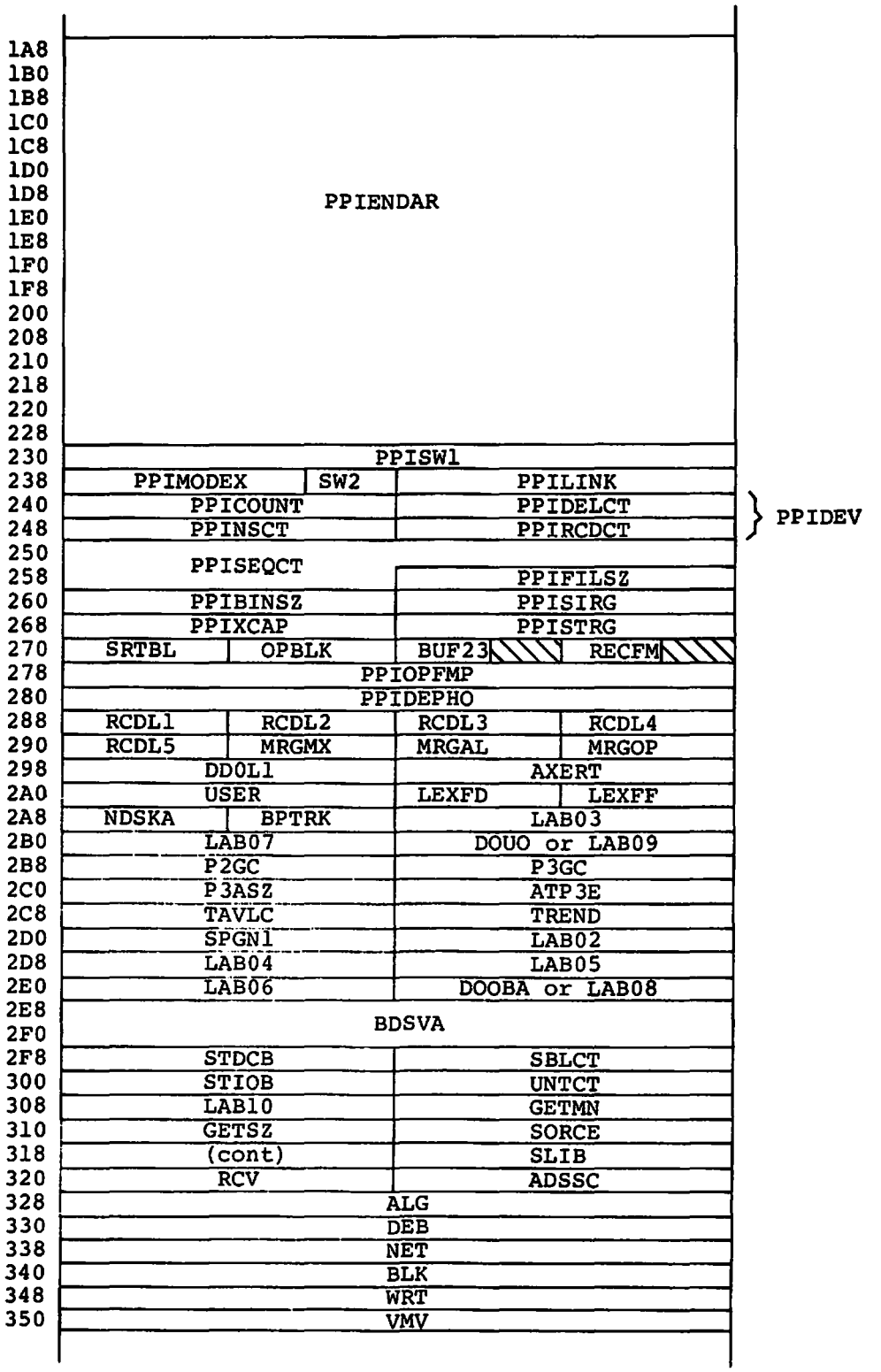


Figure 15. (2 of 3) Map of PPI

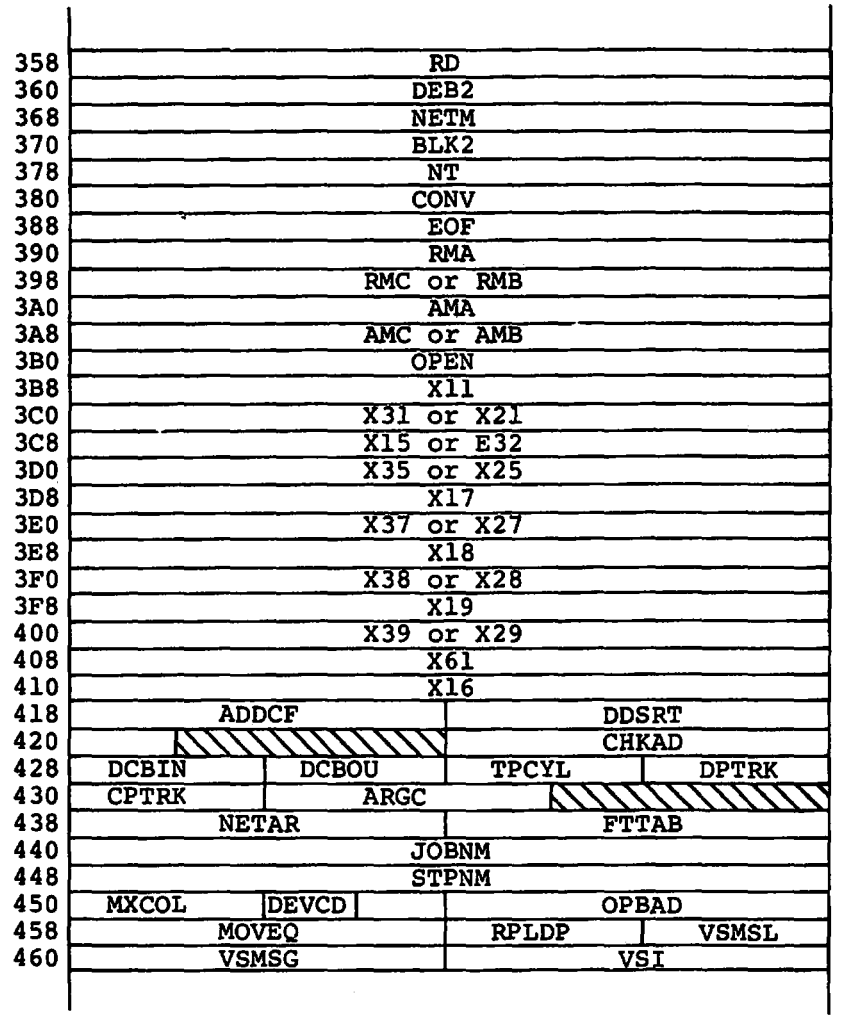


Figure 15. (3 of 3) Map of PPI

DISPLACEMENT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
0	(0)	1		X'00'
1	(1)	3	PPISM	CH'SM1' (program identifier)
72	(48)	256	PPIWKARE	Program work area (sort)
72	(48)	64	PPIPDWA	program work area (merge)
72	(48)	4	PPIP1GC	Size of Ph1 work area
76	(4C)	4	PPIP1ASZ	Ph1 assignment size
80	(50)	4	PPISKPRD	Count of skipped records
84	(54)	4	PPIATP1E	Address of invoking program's Ph1 exit
88	(58)	4	PPIIPBLK	Input blocking
92	(5C)	4	PPIBUF1	No of Ph1 buffers
96	(60)	4	PPIPGCSZ	Size of area used by extract routine
100	(64)	4	PPIFFF	Displacement of F field (used by extract routine)
104	(68)	4	PPIPBUFF	Displacement of packing buffer
108	(6C)	4		Reserved
112	(70)	72		Control Field Information
112	(70)	2	PPINUMCF	No of control fields
114	(72)	3	PPIPCF01	Control field 1: - Position
117	(75)	2	PPIMCF01	- Length
119	(77)	1	PPIFCF01	- Format + sequence
120	(78)	3	PPIPCF02	Control field 2: - Position
123	(7B)	2	PPIMCF02	- Length
125	(7D)	1	PPIFCF02	- Format + sequence
126	(7E)	3	PPIPCF03	Control field 3: - Position
129	(81)	2	PPIMCF03	- Length
131	(83)	1	PPIFCF03	- Format + sequence
132	(84)	3	PPIPCF04	Control field 4: - Position
135	(87)	2	PPIMCF04	- Length
136	(88)	64	PPIPSVA	Merge network prime area
137	(89)	1	PPIFCF04	- Format + sequence
138	(8A)	3	PPIPCF05	Control field 5: - Position
141	(8D)	2	PPIMCF05	- Length
143	(8F)	1	PPIFCF05	- Format + sequence
144	(90)	3	PPIPCF06	Control field 6: - Position
147	(93)	2	PPIMCF06	- Length
149	(95)	1	PPIFCF06	- Format + sequence
150	(96)	3	PPIPCF07	Control field 7: - Position
152	(98)	48	PPIDSKED	Used for up to 17 disk addresses, Ph1,2,3. Format: Assignment mod: ABCC (A=channel address B=unit address CC=no of tracks) Running mod: M BB CC HH R
153	(99)	2	PPIMCF07	Control field 7: length
155	(9B)	1	PPIFCF07	- Format + sequence
156	(9C)	3	PPIPCF08	Control field 8: - Position
159	(9F)	2	PPIMCF08	- Length
161	(A1)	1	PPIFCF08	- Format + sequence
162	(A2)	3	PPIPCF09	Control field 9: - Position
165	(A5)	2	PPIMCF09	- Length
167	(A7)	1	PPIFCF09	- Format + sequence
168	(A8)	3	PPIPCF10	Control field 10: - Position
171	(AB)	2	PPIMCF10	- Length
173	(AD)	1	PPIFCF10	- Format + sequence
174	(AE)	3	PPIPCF11	Control field 11: - Position
177	(B1)	2	PPIMCF11	- Length
179	(B3)	1	PPIFCF11	- Format + sequence

DISPLACEMENT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
180	(B4)	3	PPIPCF12	Control field 12: - Position
183	(B7)	2	PPIMCF12	- Length
185	(B9)	1	PPIFCF12	- Format + sequence
186	(BA)	14		Reserved
200	(C8)	64	PPILAB01	General work area (used mainly by read/write)
244	(F4)	4	PPITPPT	Tape table pointer
248	(F8)	136	PPITPTBL	Tape Table
				Two-byte entry for each unit:
		1... 0000	(Byte 1)	Input (0=output)
		.1.. 0000		Open routine should open unit
		..1. 0000		OSCL: unit contains full reel
		...1 0000		OSCL: descending sequence (set at EOF or RMAX)
		xxxx xxxx	(Byte 2)	DCB increment
280	(118)	8	PPIDIRAD	Disk directory address
288	(120)	64	PPIODOM	OSCL: odometer table
				Disk starting addresses
288	(120)	136	PPISTAR	- one entry for each of three to 17 extents
424	(1A8)	32	PPIENDAR	BALN (disk) & CRCX (Ph0,1,2): end addr reset from PPIDSKED
552	(228)	8		CRCX (Ph3): disk start addr (for read priming)
560	(230)	8	PPISW1	Switch of 64 Bits
		1...	(Byte 1)	Fix
		.1..		Var
		..1.		No mult.
		...1		Mult.
		.... 1...		BALN (tape or disk)
		.... .1..		POLY
		.... ..1.		OSCL
		x		Reserved
		x...	(Byte 2)	Reserved
		.1..		Tape work areas
		..1.		BALN (disk)
		...1		No data chaining
		.... 1...		Data chaining, input
		.... .1..		Data chaining, output
		.... ..1.		No exits to be used
		1		Exits to be used
		1...	(Byte 3)	No records over 256 bytes
		.1..		Records over 256 bytes
		..1.		SKIPREC option used
		...1		Ph1 in progress or complete
		.... 1...		Ph2 in progress or complete
		.... .1..		Ph3 (sort) in prog or compl
		.... ..1.		Ph3 (merge) in prog or compl
		1		Checkpoints to be taken
		1...	(Byte 4)	Mult.
		.1..		Extract
		..1.		SORTOUT in descending order (0=ascending)
		...1		Order in Ph1 (or merge input) is descending (0=ascending)

DISPLACEMENT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
		.... 1...		Descending order for merge pass (0=ascending)
		.... .1..		SM1 invoked (0=executed)
		.... ..1.		FILSZ or SIZE given
		1		Merge-only: input present OSCL: QSAM has detected EOF
		1...	(Byte 5)	'E'-type ctl fields present
		.1..		SORT statement present
		..1.		MERGE statement present
		...1		RECORD statement present (or, from ICERCI onwards, disk work devices with RPS)
		.... 1...		MODS statement present
		.... .1..		RECORD stmt absent or wrong
		.... ..1.		Buffers doubleword aligned
		1		Buffers fullword aligned
		1...	(Byte 6)	Read error
		.1..		Write error
		..1.		BALN (tape): odd-numbered pass (0=even)
		...x x...		Channels available: 00 - one multiplexor 01 - one multiplexor and/or one selector 10 - one multiplexor and/or several selector
		.... .1..		Input unit used as work unit
		.... ..1.		Tape switch or alternative channel available
		1		More than 1 channel available
		1...	(Byte 7)	Deblock backward
		.1..		Read forward
		..1.		Close with rewind
		...1		Block forward
		.... 1...		Read forward later
		.... .x..		Reserved
		.... ..1.		CRCX
		1		DIAG option being used
		1...	(Byte 8)	OSCL: user EOF
		.1..		OSCL: RMAX reached
		..1.		OSCL: user insert in process
		...1		Some or all tapes are 7-track
		.... 1...		CRCX: merge pass to follow
		.... .1..		Sort called by IMS
		.... ..1.		Up to ICERCN: max. no. of control fields present From ICERCN: BALN
		1		User rtn has given EROPT=skip/accept
568	(238)	3	PPIMODEX	Exits Activated
		1...		E11
		.1..		E15 or E32
		..1.		E16
		...1		E17
		.... 1...		E18
		.... .1..		E21
		.... ..1.		E25
		1		E27

DISPLACEMENT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
		1...	(Byte 2)	E28
		.1..		E31
		..1.		E35
		...1		E37
		.... 1...		E38
		.... .1..		E61
		.... ..1.		E19
		1		E29
		1...	(Byte 3)	E39
		.xxx x...		Reserved
		.... .1..		RECFM=FB in SYSOUT DCB
		.... ..1.		Input is spanned records
		1		Output is spanned records
571	(23B)	1	PPISW2	Switch of 8 Bits
		1...	PPIFSEON	FILSZ given with exact value
		.1..		Exec sort with ABEND option
		..1.		AQ control fields present
		...1		Preserve order of equals
		.... 1...		VSAM input
		.... .1..		KSDS VSAM output
		.... ..1.		ESDS VSAM output
		1		SIZE(MAX) used
572	(23C)	4	PPILINK	Link Edit Information for exit rtns (set by RCH)
		xxxx xxxx	(Byte 1)	Exits (same order as PPIMODEX) :
		xxxx xxxx	(Byte 2)	1=Routine link edited,
		x...	(Byte 3)	0=Not edited (via the program)
		.1..		E11 routine edited separately
		..1.		E21 routine edited separately
		...1		E31 routine edited separately
		.... 1...		Linkage editing carried out
		.... .1..		VSAM output not reusable
		.... ..1.		Linkage edit error
		1		VSAM release 1 in system
		xxxx xxxx	(Byte 4)	Reserved
576	(240)	16	PPIDEV	Device inf: words 2-5 fr DEVTYPE macro used by ICERCI
576	(240)	4	PPICOUNT	Count of records: Ph1 - deblocked from input Ph2 - written to work units Ph3 - placed on SORTOUT
580	(244)	4	PPIDELCT	Count of records deleted
584	(248)	4	PPIINSCT	Count of records inserted
588	(24C)	4	PPIRCDCT	Count of records entering ph.
592	(250)	12	PPISEQCT	BALN (disk) : count of strings per work area (halfword/area) BALN (tape) : count of strings-
			(Bytes 1-4)	- entering phase
			(Bytes 5-8)	- written out
604	(25C)	4	PPIFILSZ	Value of FILSZ or SIZE param.
608	(260)	4	PPIBINSZ	Bin size
612	(264)	4	PPISIRG	Size of interblock gap (tapewk)
612	(264)	4	PPISRALG	Addr of sector conversion rtn (RPS)
616	(268)	4	PPIXCAP	OSCL: RMAX (max no. bytes/reel) Otherwise: Cap. of interm. storage, in records
620	(26C)	4	PPISRTG	G (Cap. of RSA, in records)
624	(270)	2	PPISRTBL	B (blocking used on wk areas)
626	(272)	2	PPIOPBLK	Output blocking

DISPLACEMENT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
628	(274)	2	PPIBUF23	No of buffers (Ph2,3)
630	(276)	1	PPIRECFM	User record format for Ph3
631	(277)	1		Reserved
632	(278)	8	PPIOPFMP	Output unit for Ph3
640	(280)	8	PPIDEPHO	Output unit address
648	(288)	2	PPIRCDL1	l_1 from RECORD stmt (LENGTH)
650	(28A)	2	PPIRCDL2	l_2 from RECORD stmt (LENGTH)
652	(28C)	2	PPIRCDL3	l_3 from RECORD stmt (LENGTH)
654	(28E)	2	PPIRCDL4	l^4 from RECORD stmt (LENGTH)
656	(290)	2	PPIRCDL5	l^5 from RECORD stmt (LENGTH)
658	(292)	2	PPIMRGMX	Maximum merge order
660	(294)	2	PPIMRGAL	BALN & POLY: alternate merge order
662	(296)	2	PPIMRGOP	Disk sort: optimum merge order
664	(298)	4	PPIDD0L1	Ph2,3: length & offset of control field 1
668	(29C)	4	PPIAXERT	Addr of mult. or extract rtn
672	(300)	4	PPIUSER	User communication area
676	(2A4)	2	PPILEXFD	Length of extracted fields
678	(2A5)	2	PPILEXFF	PPILEXFD rounded up to fullwd
680	(2A8)	2	PPINDSKA	No of disk work areas
682	(2AA)	2	PPIBPTRK	Blocks/track for disk wk areas
684	(2AC)	4	PPILAB03	Input buffer information:
			(Byte 1)	No of buffers
			(Bytes 2-4)	Size of buffers
688	(2B0)	4	PPILAB07	Output buffer information:
			(Byte 1)	No of buffers
			(Bytes 2-4)	Size of buffers
692	(2B4)	4	PPIDOUO	X'04'=suppress sequence check (passed by user from E35)
692	(2B4)	4	PPILAB09	Used to alloc. output buffers:
			(Byte 1)	No of Ph2 output buffers
			(Byte 2)	No of Ph3 output buffers
			(Bytes 3-4)	Size of Ph3 output buffers
696	(2B8)	4	PPIP2GC	Size of Ph2 work area
700	(2BC)	4	PPIP3GC	Size of Ph3 work area
704	(2C0)	4	PPIP3ASZ	Message option
708	(2C4)	4	PPIATP3E	Addr of invoking pgm's Ph3 rtn
712	(2C8)	4	PPITAVLC	Size of available core, Ph1
716	(2CC)	4	PPITREND	Ending address of tree
720	(2D0)	4	PPISPGN1	Address of next byte available in phase work area
724	(2D4)	4	PPILAB02	Address of input buffer table OSCL: address of Ph1 input buffer table
728	(2D8)	4	PPILAB04	Address of output buffer 1
732	(2DC)	4	PPILAB05	Address of output buffer 2
736	(2E0)	4	PPILAB06	Address of control buffer - Ph1: input buffer pool Merge-only Ph3: out.buff. pool
740	(2E4)	4	PPIDOOBA	Addr of record for Ph3 sequence check
740	(2E4)	4	PPILAB08	RSA Table information:
			(Byte 1)	No of entries
			(Bytes 2-4)	Address of table
744	(2E8)	16	PPIBDSVA	Block/deblock area:
			(Byte 1)	No of work units
			(Bytes 2-4)	Addr of inp.buff. table, Ph2&3
			(Bytes 5-8)	Fix: addr of Ph1 move list Var: addr of next avail. bin

DISPLACEMENT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
			(Bytes 9-12)	Fix: addr of Ph2&3 move list Var: no of available bins
			(Bytes 13-16)	Var: entry to Ph1 move rtn
760	(2F8)	4	PPISTDCB	Address of DCB table
764	(2FC)	4	PPISBLCT	Address of block count table
768	(300)	4	PPISTIOB	Address of IOB table
772	(304)	4	PPIUNTCT	OSCL: addr of unit count table
776	(308)	4	PPILAB10	OSCL, Ph2: addr of input buffer table
780	(30C)	4	PPIGETMN	Addr of GETMAIN table of addr.
784	(310)	4	PPIGETSZ	Addr of GETMAIN table of sizes
788	(314)	8	PPISORCE	Up to RCZ: DDname of user mod library RCZ: DCB addr of SYSLMOD user user lib (bytes 1-4)
796	(31C)	4	PPISLIB	DCB address of sort library
800	(320)	4	PPIRCV	Reserved
804	(324)	4	PPIADSSC	Return address to ICERCV
808	(328)	240		Module Interface List
808	(328)	8	PPIALG	Ph1,2: Algorithm
816	(330)	8	PPIDEB	Ph1,3: Deblock
824	(338)	8	PPINET	Ph1,3: Network
832	(340)	8	PPIBLK	Ph1,3: Block
840	(348)	8	PPIWRT	Ph1,2: Write
848	(350)	8	PPIVMV	Ph1: Move (var)
856	(358)	8	PPIRD	Ph2,3: Read
864	(360)	8	PPIDEB2	Ph2: Deblock (prime rtn)
872	(368)	8	PPINETM	Ph2: Network
880	(370)	8	PPIBLK2	Ph2: Block/deblock
888	(378)	8	PPIINT	OSCL: Initialize sort and tree
896	(380)	8	PPICONV	Convert hex to char for msgs
904	(388)	8	PPIEOF	Ph1,3 (merge-only) : EODAD for QSAM
912	(390)	8	PPIRMA	Messages, Ph1 running modules
920	(398)	8	PPIRMB	Messages, Ph2 running modules
920	(398)	8	PPIRMC	Messages, Ph3 running modules
928	(3A0)	8	PPIAMA	Messages, Ph1 assignmt modules
936	(3A8)	8	PPIAMB	Messages, Ph2 assignmt modules
936	(3A8)	8	PPIAMC	Messages, Ph3 assignmt modules
944	(3B0)	8	PPIOPEN	Ph1,2,3: Open list
952	(3B8)	8	PPIX11	Exits for
960	(3C0)	8	PPIX21	functions
960	(3C0)	8	PPIX31	user initialization
968	(3C8)	8	PPIX15|X32	Exits for
976	(3D0)	8	PPIX25	modification
976	(3D0)	8	PPIX35	logical record
984	(3D8)	8	PPIX17	Exits for
992	(3E0)	8	PPIX27	at end of phase
992	(3E0)	8	PPIX37	closing user data sets
1000	(3E8)	8	PPIX18	Exits
1008	(3F0)	8	PPIX28	errors
1008	(3F0)	8	PPIX38	for read
1016	(3F8)	8	PPIX19	Exits
1024	(400)	8	PPIX29	errors
1024	(400)	8	PPIX39	for write
1032	(408)	8	PPIX61	Exit to modify control fields
1040	(410)	8	PPIX16	Exit when NMAX exceeded
1048	(418)	4	PPIADDCF	When more than 12 ctl fields: addr of additional information

DISPLACEMENT DEC	(HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
1052	(41C)	4	PPIDDSRT	4 chars to replace SORT in DDnames (from param list when the program invoked)
1056	(420)	1		PPI version number
1057	(421)	3		Reserved
1060	(424)	4	PPICHKAD	Checkpoint module address
1064	(428)	2	PPIDCBIN	Size of SORTIN DCB
1066	(42A)	2	PPIDCBOU	Size of SORTOUT DCB
1068	(42C)	2	PPITPCYL	No of trks/cyl. (disk wk areas)
1070	(42E)	2	PPIDPTRK	No of directory blocks/track (disk work areas)
1072	(430)	2	PPICPTRK	Track capacity in bytes
1074	(432)	3	PPIAPGC	Addr of permanently gen. core
1077	(435)	3		Reserved
1080	(438)	4	PPINETAR	Addr of network table in phase work area
1084	(43C)	4	PPIFTTAB	BALN (disk): pointer to table of formatted tracks
1088	(440)	8	PPIJOBNM	Jobname
1096	(448)	8	PPISTPNM	Stepname
1104	(450)	2	PPIMXCOL	Rightmost col included in any control field
1106	(452)	1	PPIDEVCD	Device code for unit with RPS
1107	(453)	1		Reserved
1108	(454)	4	PPIOPBAD	Addr of record in output buffer for Ph2 sequence check
1112	(458)	4	PPIMOVEQ	Addr of 2nd move rtn for equals
1116	(45C)	2	PPIRPLDP	RPL displ. from ACB
1118	(45E)	2	PPIVSMSL	Length of generated msg area
1120	(460)	4	PPIVSMSG	Addr of general VSAM msg area
1124	(464)	4	PPIVSI	Ph1 addr of VSAM read module

Index to PPI

The following index covers PPI fields only. For any given subject, it shows the name of the PPI field referenced, plus the field's displacement from the beginning of the PPI area, in hexadecimal.

Since all fields begin with the letters 'PPI', these letters are omitted.

When the reference is to a field which occupies a whole number of bytes, the displacement is shown in whole bytes. When the reference is to one or more bits within the field, the reference is given in the form 'bytes.bits'. For example, the field PPISW1 is at displacement 560. A reference to 'SW1,561.0' therefore means the first bit (bit 0) of the second byte in the field.

Module ICEAM1: Generated Defaults

The option values generated when the program is installed are stored in module ICEAM1. The format of the module is shown below. Values underlined are generated by default if no specification is made when the program is installed. The generated values can be changed by use of the IMASPZAP, HMASPZAP (VS1) or AMASPZAP (VS2) Service Aid Program.

DISPLACEMENT DEC	DISPLACEMENT (HEX)	SIZE bytes	FIELD NAME	CONTENTS
0	0	4	SIZE	X'00FFFFF8' (representing MAX) or size value in hexadecimal
4	4	4	MAXLIM	Representing MAXLIM value in hexadecimal (Default: X'00080000')
8	8	4	MINLIM	MINLIM value in hexadecimal (Default: X'00012000')
12	C	4	RESALL	All disk sorts reserved main storage value in hexadecimal (Default: X'00001000')
16	10	4	RESINV	Invoked sorts reserved main storage value in hexadecimal (Default: 0)
20	14	4	MSGS	'NYYN' (FLAG(I)) 'NYNY' (FLAG(U)) 'YNNY' (NOFLAG) 'YNYN' (AC)
24	18	1	EQUALS	'Y' (YES) 'N' (NO)
25	19	1	LIST	'Y' (YES) 'N' (NO)
26	1A	1	ERETJCL	'Y' (ABEND) 'N' (RC16)
27	1B	1	ERETINV	'Y' (ABEND) 'N' (RC16)
28	1C	8	PRINT	Message DDname for invoked sort (Default: 'SYSOUT')

DISPLACEMENT DEC	(HEX)	SIZE bytes	FIELD NAME	CONTENTS
36	24	1	VIO	'Y' (YES) 'N' (NO)
37	25	1	EXCPVR	'Y' (YES) 'N' (NO)
38	26	1	RESDNT	'Y' (ALL) 'M' (MON) 'N' (NONE)
39	27	1	RELEASE	'Y' (YES) 'N' (NO)
40	28	1	SECALL	'Y' (YES) 'N' (NO)
41	29	1	VERIFY	'Y' (YES) 'N' (NO)
42	2A	1	BLKSET	'Y' (YES) 'N' (NO)
43	2B	1	CHALT	'C' (YES) 'A' (NO)
44	2C	2	SVC	X'0A6D' (the operations code for SVC 109 or SVC 200 through 255 in hexadecimal)
46	2E	1	CHECK	'Y' (YES) 'N' (NO)
47	2F	1	SMF	'F' (FULL) 'S' (SHORT) 'N' (NO)
48	30	1	VBLKSET	'Y' (YES) 'N' (NO)
49	31	256	ALTSEQ	Translate table for AQ control fields (Default: standard EBCDIC sequence)

Note: If you have changed the generated values and reenterable modules are placed resident, you have to re-IPL the operating system to activate the changes. For SVS/MVS specify option CLPA when you re-IPL.

Section 6. Debugging Aids

This section is intended to help you if Sort behaves in an unexpected way, and you want to localize the problem and if possible solve or bypass it. It contains the following subsections:

- How to decide whether a problem is due to a program error.
- General considerations when an error has been located.
- How the microfiche are organized.
- Adding a temporary change to the maintenance area.
- How Sort uses registers.
- How to read a dump to find DCBs and IOBs for SORTIN, SORTOUT, and SORTWK.
- Various uses of the DEBUG control statement.
- Explanations of the diagnostic messages.
- Explanation of program dumps.
- Finding an object module in a storage dump.
- Message-to-module cross reference. How to find messages in the code.
- Cross references showing the displacement of each field in the common data areas (COMMON, COMMA, CPI, and PPI) for the various techniques.

Other diagnostic aids are provided in Section 3, where the phase structures are given; Section 5, which shows the layout of some important control blocks; and Appendix A, showing which modules activate the various program exits.

Defining Problem Cause

If the program is unable to successfully complete sorting or merging, you will certainly get one or more program messages, and possibly also an ABEND code.

The OS/VS Sort/Merge Programmer's Guide gives explanations of the various program messages, and suggestions as to how to cope with them. It is assumed that you have exhausted those explanations before turning to this section.

IS THIS A PROGRAM ERROR?

Your first task is to decide whether or not the problem is due to an error in Sort code.

If your installation has just installed a new release or PTF level of Sort, it is worth checking that any necessary additional alias names have been added to module ICEMAN. If they have not, mixed levels of program modules can be executed, which can give rise to unpredictable abnormal terminations.

Otherwise, if Sort is run alone in its region, problems are unlikely to arise from the environment. If no routines of yours were invoking Sort, or being used at program exits, you can therefore work on the assumption that you have found a program error, and turn to the section 'Bypassing the Problem' below.

However, if you are invoking Sort from a program of your own, or if you are using routines at program exits, you will need to eliminate your own programs as sources of error. In the example in Figure 16, for instance, an exit (E15) is being used.

```
ICE000I   --- CONTROL STATEMENTS/MESSAGES ---- 5740-SM1 REL 4.0 ...

           SORT FIELDS=(1,5,CH,A),EQUALS
           RECORD TYPE=F,LENGTH=(1200,,1000)
           MODS E15=(E15,79000,MODSLIB,N)
ICE074I   RECORD LENGTH L1 OR L3 OVERRIDDEN
ICE088I   SORTJOB.SORTSTEP, INPUT LRECL=1200, BLKSIZE=12000, TYPE=F
ICE093I   MAIN STORAGE = (MAX,262144,48528), NMAX=7300, PEERAGE
ICE039A   INSUFFICIENT MAIN STORAGE - ADD 6K BYTES
```

Figure 16. A Sample Set of Messages

POTENTIAL PROBLEMS WITH ROUTINES AT PROGRAM EXITS

Use of Registers

The first thing to check with your routines is that they observe the standard linkage conventions. If, for example, they corrupt register 12, results are unpredictable but almost certain to result in an ABEND of some kind, because Sort uses register 12 as its main base register.

Check, too, that you are not using registers for loading or storing that are accidentally causing overlay of sort code or work areas. If this happens Sort could work without errors with one technique, but fail with another.

Space

The next thing to check is whether your routines are trying to use more space than you have allocated to them. Have you installed a new operating system release since the last time you used these routines? Each time you use an OPEN macro, for example, your program may take buffer space; but the amount it tries to take will depend upon such factors as the current release of the operating system.

A change of operating system could therefore lead to an ABEND in your own routine; or it could lead to too little space being left for Sort.

You can see whether too little space was left for sorting by studying the information in message ICE093I. As shown in Figure 16, the third field in the 'MAIN STORAGE' parameter will tell you how much was actually left for Sort after your own routines had taken what they needed, in a region or partition of only 256K. Since in the example the input block size is large, the 48,528 bytes left are not enough.

Similar situations can occur if Sort is dynamically invoked using the MAX option, and a fairly large reserved value is passed to Sort or taken by default.

Another problem could arise if the E15 routine issues a GETMAIN without a corresponding FREEMAIN at the end. This problem can be caused indirectly, for example by leaving a data set open so that a buffer pool remains reserved.

Record Contents

If the output records do not appear to contain the same data as the input records, and either E15 or E35 has been used, check that your routine is handling register 1 correctly; especially, check that it is correct on return to Sort.

If for example you first load register 1 and then restore all registers (including register 1), it will probably have the wrong contents.

Equally, if you first restore all registers and then try to load register 1 from a corrupted base register, you will almost certainly pass the wrong information to Sort.

POTENTIAL PROBLEMS WITH INVOKING PROGRAMS

Space can also be a problem when you invoke Sort from another program, especially if you are using SIZE=MAX and invoking exit E15 or E35 (or, from COBOL, using an Input or Output procedure).

If you do this, and particularly if you open a file in your exit routine, check that you specify a sufficiently large amount of reserved storage.

Considerations when an Error has been Located

When you have pinpointed an error to a particular module, it is always worth doing a RETAIN search to find out whether the problem is already known. If the problem itself has not been reported, you can search for other APAR fixes to the failing module, and apply them if appropriate.

If this procedure fails, you have the choice between trying to fix the problem, and trying to bypass it. You will then also want to report the problem.

WHEN NOT TO WASTE TIME ON REPAIR

Some modules contain extremely complex code. Attempts at field repair can be expected to take an unreasonable amount of time.

The following load modules are particularly unsuitable candidates for field repair:

ICEIPUT	ICEIPVT	ICEOPUT	ICEOPVT
ICEKPUT	ICEKPVT	ICEPAR	
ICEKPUS	ICEKPVS	ICERED	

BYPASSING A PROBLEM

The simplest way of bypassing a problem in the program is to force it to use a different technique. Message ICE092I or ICE093I will tell you which sorting technique has been used.

You can use the DEBUG control statement, described later in this section, to force the use or non-use of a specific technique. Alternatively, if the problem is in the Blockset technique (the newest in the program), you can 'turn off' that technique altogether by including an OPTION statement with the parameter NOBLKSET.

REPORTING A PROBLEM

You have two alternative means available for reporting a problem: an incident report, or an APAR.

Submitting an Incident Report

Generally, central service needs the following information in a RETAIN incident entry:

- Symptom
- Was SORTIN or SORTOUT DD statement present? If so, list the DCB attributes.
- Was Sort invoked from another program? If so, try to list the contents of the parameter list passed. See the OS/VS Sort/Merge Programmer's Guide for parameter list format. At entry to Sort register 1 must point to the parameter list. Note, however, that (except with a standard disk sort) the initial contents of register 1 are lost towards the end of Phase 0.
- Was Sort invoked via JCL? If so, list all the program control statements used.
- List Release and PTF levels, and if possible the APAR fixes applied since the last PTF.
- Give the number of SORTWK data sets used, and their unit type.
- If message ICE092I or ICE093I was issued, give its contents.
- If you have a dump, list the Sort module(s) in storage when the program failed, and the displacement into the failing module at which the abend occurred.

Submitting an APAR

In addition to the information requested on an APAR form, you should supply the following information:

- A dump from the failing run of Sort.
- A list of PTFs applied, and if possible the APAR fixes applied since the last PTF.
- Source code (Assembler or compiler listings) and object code for any user routines and/or invoking program.
- A listing of the JCL.
- A copy of the input file(s) in machine-readable form.
- A list of program control statements used.

- A list of the generation options selected, which can be produced by using IMASPZAP (or an equivalent program) to dump the contents of module ICEAM1.

Microfiche Organization

For Peerage and Vale each load module is on one fiche, with the same name as the load module name. For example, module ICEDEF is completely contained on a fiche called ICEDEF.

Blockset has several fiches for each load module. Each object module is on a separate fiche. For example, load module ICEMESI consists of five object modules (ICEMESI, ICEMESS, ICEEXII, ICEMSGI, and ICESUBS), and is on five fiches with those names, as shown in Figure 17.

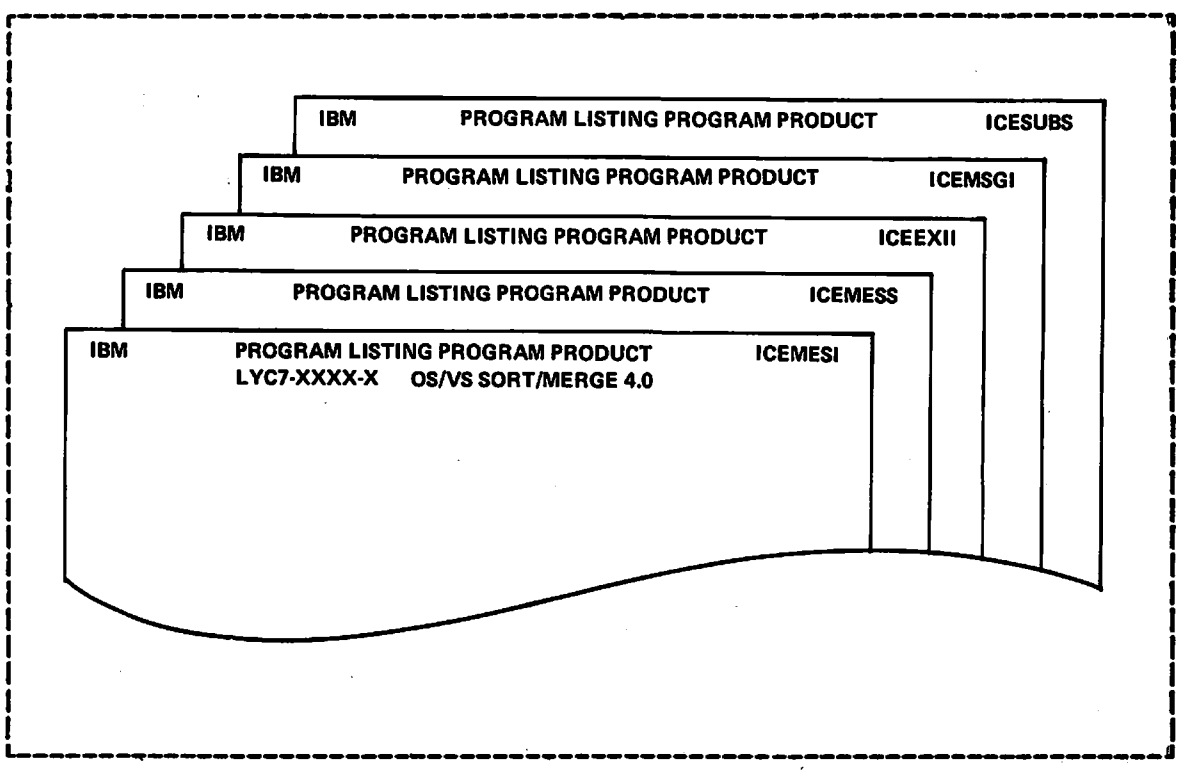


Figure 17. Microfiche for Load Module ICEMESI

The other techniques have one fiche per load module, except for modules ICERCM and ICERCZ. These two modules have an overlay structure, and are shown with one fiche per object module. Section 3 gives further details of module organization under the heading 'Phase Structures'.

Adding a Temporary Change to the Maintenance Area

Every new Sort module has a maintenance patch area.

For Peerage and Vale the area is included in each load module. It is 100 bytes long, or 5% of the size of the module, and is at the end of the module. It is usually addressable by using register 11 plus the appropriate displacement (which can be calculated from the microfiche), and generally starts with the label PATCH in the source code.

Blockset has a maintenance patch area of at least 64 bytes in each object module. It can be found in the fiche by looking for the label PATCH.

The other, older techniques do not have patch areas. Instead you can use the EXPAND facility of the linkage editor to create one at the end of a module, if required.

To use a patch area, first verify that it contains all zeros. Then use the appropriate utility (IMASPZAP, HMASPZAP, or AMASPZAP) to put in the required change.

If the Sort reenterable modules are in the LPA library, do not forget to re-IPL with the CLPA option.

How Sort Uses Registers

In addition to the conventions described in Appendix D, the Peerage/Vale techniques make extensive use of certain internal register conventions. They are shown in Figure 18.

Register	Use
0 - 5	Work registers
6	Current RSA pointer
7	Pointer to current record in input buffer for SORTIN or SORTWK
8	Pointer to current record in output buffer for SORTWK or SORTOUT
10	Internal return register when branch-and-link to sub-routines is used
11	Secondary base register
12	Primary base register
13	Pointer to the common area (COMMA or COMMON)
14 - 15	Work registers

Figure 18. Use of Registers by Peerage/Vale Techniques

Finding DCBs and IOBs for SORTIN, SORTOUT, SORTWK

This subsection describes how to locate the I/O control blocks in a system dump, for the standard disk techniques.

The first step is to verify, from message ICE092I or ICE093I, which of the standard techniques has been used.

You should then find the common area. Look at the address pointed to by register 13, and check whether the first word contains the characters 'SM1'. If it does, register 13 is intact, and can be used when calculating offsets. If it does not, and you are certain the Sort program is being used, then register 13 has been corrupted. However, you can still find the common area by looking for the eyecatcher 'SM1' with which it begins.

BLOCKSET TECHNIQUE

SORTIN and SORTOUT Control Blocks

Figure 19 shows an example of interpreting a system dump for Blockset.

The procedure for finding input and output control blocks is as follows:

1. Look in COMMON at label COMINDCB, which is at register 13 + X'154'. There you will find a pointer to the SORTIN DCB.
2. The SORTOUT DCB pointer is at label COMUTDCB, at register 13 + X'158'.
3. There is no pointer from the DCBs to the corresponding IOBs, because EXCP is used for access. However the IOBs are in the same areas as the corresponding DCBs, at displacement X'48' from the start of the DCB.
4. A fullword at IOB + X'14' points back to the DCB.

SORTWK Control Blocks

The procedure for finding SORTWK control blocks is as follows:

1. Look in COMMON at label COMWKDCB, at register 13 + X'15C'. This is a pointer to the DCB list.
2. Follow the pointer to the SORTWK DCB list. Each element in the list is a pointer to a DCB. The last element in the list has the first bit on.
3. As for SORTIN and SORTOUT, the access method used is EXCP. The IOB for each work area follows the corresponding DCB, at DCB start address + X'48'.

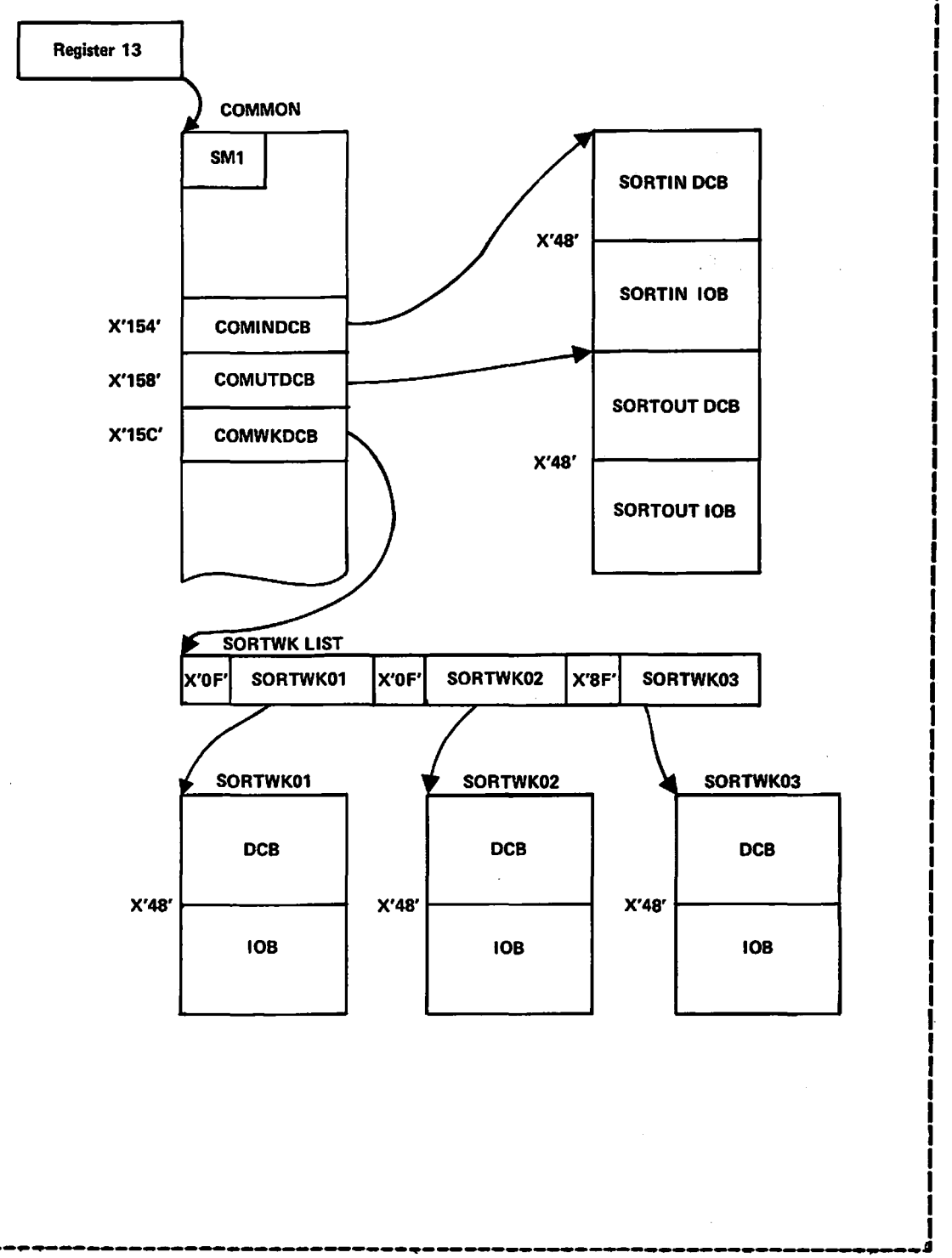


Figure 19. Locating Control Blocks in a Dump with Blockset

PEERAGE AND VALE TECHNIQUES

The layout of all Peerage and Vale internal work I/O control blocks can be found in the microfiche for modules ICEMON and ICEDEV.

SORTIN and SORTOUT Control Blocks

First check which access method has been used. If EXCP was used, you will have received message ICE084I; otherwise (if SAM was used) you will not.

1. Field CSORTIN in COMMA, at register 13 + X'5FC', points to the SORTIN DCB.

2. Field CSORTOUT, at register 13 + X'600', points to the SORTOUT DCB.

3. If EXCP is used, most I/O information is in a storage area, subpool 0, with label OBSBLOCK. See Figure 20. A pointer to this area is in ICECOMMA at label COBSBLOK, at register 13 + X'7E4'.

   At OBSBLOK + X'8' there is an eyecatcher consisting of the characters 'OBSBLOCK'.

   The layout of OBSBLOCK is given in the microfiche ICEOBS and ICEXCP.

   a. In one of them, find the offsets to label OBSIOBP.

   b. Add the value of the EQU' (OBSIOBP) to the address of OBSBLOCK (taken from register 13 + X'7E4', as described above). This gives the address of the IOB: if you are in Phase 1 it will be the SORTIN IOB; in Phase 3 it will be the SORTOUT IOB. If you are not sure of the phase, check the DCBs to see which one is open.

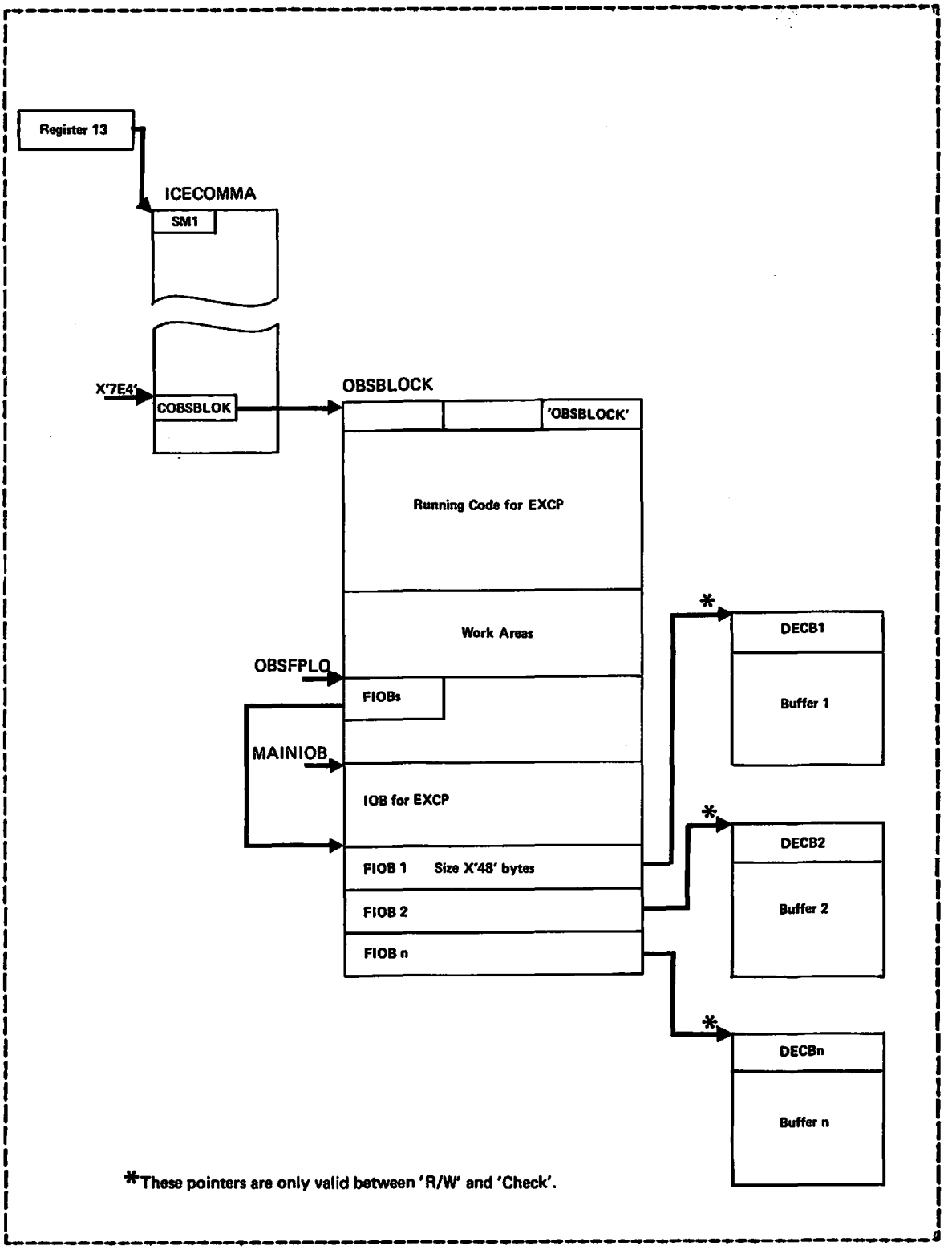


Figure 20. Locating SORTIN and SORTOUT Control Blocks with EXCP in a Dump with Peerage and Vale

4. OBSBLOCK also contains read/write and check simulators, used to simulate SAM access methods. The SORTIN and SORTOUT DCBs contain the addresses of the simulators, at DCB + X'30' and DCB + X'34' respectively.

5. After a read or write macro has been used to invoke a simulator, and before the subsequent check macro is issued, there are also valid pointers from OBSBLOCK to the DECB at the beginning of each buffer. See Figure 20. The address of the pointer list (a list of fake IOBs, or FIOBs) is at label OBSIFFIO. To find a DECB:

   a. Look in the fiche for ICEOBS or ICEXCP, and find label OBSFIOBP.

   b. Add the value of the EQU(OBSFIOBP) to the address of OBSBLOCK, taken from register 13 + X'7E4'.

   c. The result is the address of a pointer to the first FIOB.

   d. The other FIOBs follow consecutively, each X'48' bytes long on a doubleword boundary.

   e. The DCB NCP value tells how many valid FIOBs there are.

   f. The FIOB plus 4 points to the DECB.

SORTWK Control Blocks

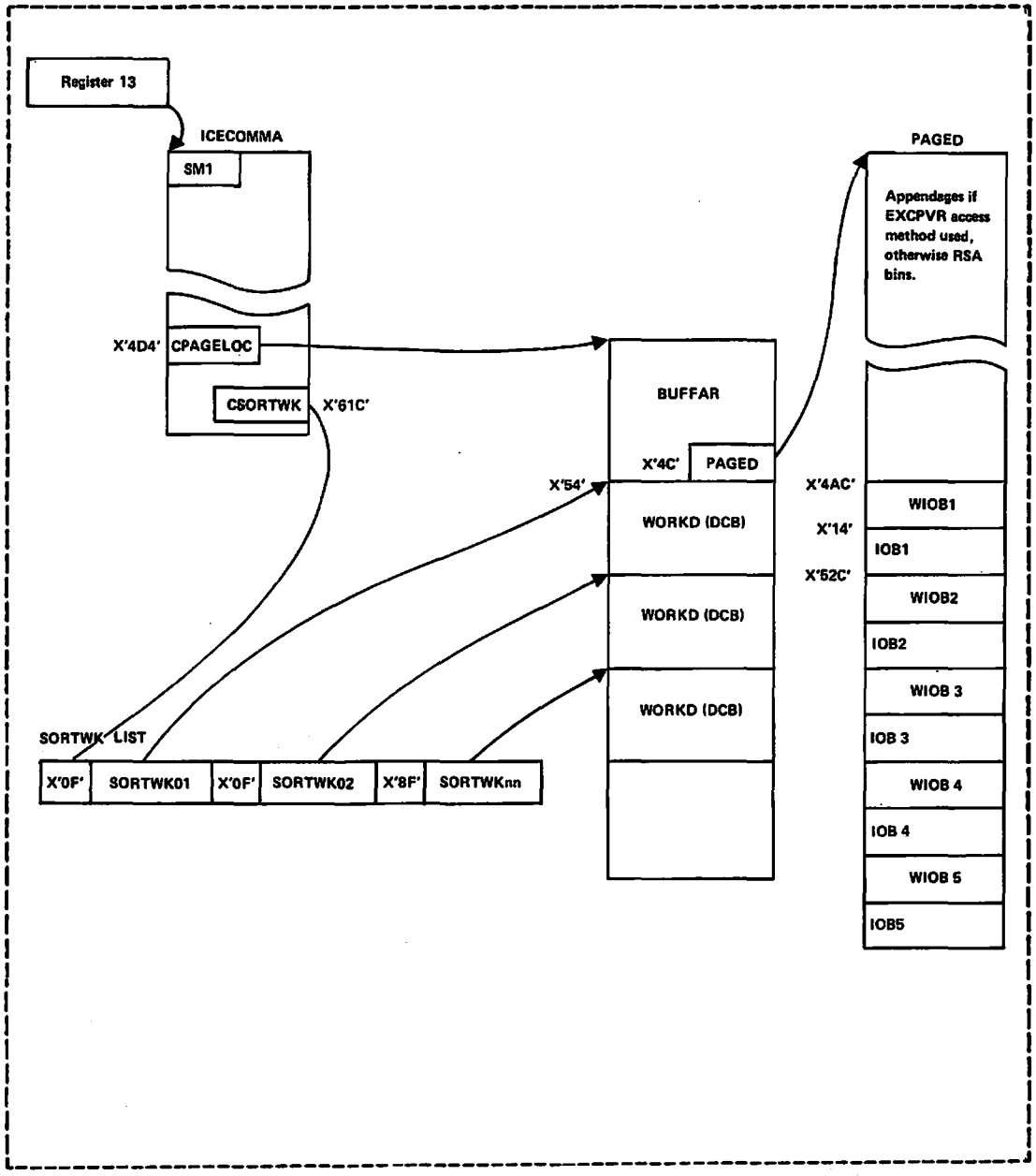


Figure 21. Locating SORTWK Control Blocks in a Dump with Peerage/Vale

The procedure to find a work IOB is as follows:

1. Look in COMMA at label CPAGELOC, at register 13 + X'4D4'. This field contains a pointer to a control block called BUFFAR.
2. Follow the pointer to BUFFAR. At BUFFAR + X'4C' you will find a pointer to a control block called PAGED.
3. Follow the pointer to PAGED. At PAGED + X'4AC' is an area, WIOB, containing five control blocks.
4. WIOB + X'14' contains the IOB. IOB + X'14' points to the DCB.
5. The WIOB control blocks are each X'80' bytes long, and follow each other consecutively.

You can also go straight to the DCB, as follows:

The field CSORTWK in COMMA, at register 13 + X'61C', contains a pointer to the SORTWK DCB list. Each word in the list points to a work DCB, which forms part of a control block called WORKD. The last word in the list has the first bit on.

The DEBUG Control Statement

This statement is only valid when the program meets the criteria for the standard disk techniques. If it is supplied under other circumstances it is ignored.

The statement is not intended for regular use; only the first two parameters are of general interest. The other parameters can be used to provide a temporary bypass, or to supply detailed information on program execution for use when optimizing or debugging the standard disk sort.

DEBUG can be passed to an invoked sort by means of the SORTCNTL DD statement, for example:

```
//SORTCNTL  DD  *
          DEBUG BSAM
```

This DD statement is described in full in the OS/VS Sort/Merge Programmer's Guide. Note that the DD name might not always be SORTCNTL, because the first four letters of Sort's special DD statement names can be changed for an invoked application. It might, for example, need to be called //TESTCNTL instead. See the chapter in the Programmer's Guide on invoking Sort from another program. The Blockset techniques will not be used when the DEBUG control statement is used with a SORTCNTL DD statement.

SYNTAX

```
[label] DEBUG  ABEND     ,DUMP    [,PEERVALE] ,BALN
               NOABEND   ,NODUMP              ,CRCX
              [,CLOCK] [,FLAG] [,CTRx] [,BSAM]
```

ABEND
NOABEND
Overrides the generated default for action to be taken when the program encounters an uncorrectable error.

DUMP and NODUMP
Recognized but ignored.

PEERVALE
With a disk sort one of the three standard techniques (Blockset, Peerage, or Vale) is normally used. If you have encountered a problem when using Blockset (see message ICE092I or ICE093I), you can temporarily bypass this technique by specifying PEERVALE, thus forcing Peerage or Vale to be used.

BSAM
With the disk sort techniques Peerage and Vale, normally the EXCP access method is used for SORTIN and SORTOUT. If you encounter a problem related to this I/O activity you can temporarily bypass it by specifying BSAM.

BALN
CRCX
With a disk sort you can use this parameter to force either the balanced (BALN) or crisscross (CRCX) disk sorting technique and therefore bypass the standard disk sort techniques used by the program. If either BALN or CRCX is forced then the following restrictions apply:

- At least three work data sets on the same type of device are needed, with amount as specified in OS/VS Sort/Merge Programmer's Guide. Mixed device types are not allowed.
- Maximum record length must be less than work device track length.
- Allocation must be contiguous (the CONTIG parameter is required), and only primary extents will be used.
- Six or more work data sets are required for the CRCX technique.
- For SORTWKnn: nn can be any number from 01 to 32. The first number must be 01 and the others must follow consecutively with no gaps.
- Unused work space will not be released; the RLSE parameter must not be specified.

CLOCK (only Peerage and Vale)	Instructs the program to measure elapsed and CPU times for the different phases.
FLAG(0) (only Peerage and Vale)	Instructs the program to print information messages (ICE120-125). These messages are listed under 'Messages Produced by Using the DEBUG Statement'.
CTRx=value	Specifying this parameter will force Peerage or Vale to be used. The program will keep a count of the input or output records. When the count reaches the value specified the program will ABEND, and a formatted dump will be printed. The numbers that may be assigned to x are: 2 Count of input records being moved from the input buffer. 3 Count of output records being moved to the output buffer. 4 Count of input records inserted by E15. 5 Count of output records deleted by E35.

Note: When the count reaches 'value', the program will ABEND. It will also terminate with message ICE025A if the 'value' is a number greater than the number of input records.

MESSAGES PRODUCED BY USING THE DEBUG CONTROL STATEMENT

Messages ICE120-125 are issued if the DEBUG statement is supplied with the appropriate parameters FLAG(2) (only for Peerage and Vale sorts).

ICE120O RL=a B=b IL=c IS=d IB=e RM=f EM=g BA=h IX=j OX=k

Explanation:

This message relates to the Optimization part of the definition phase (Phase 0).

- RL record length (within the Sort);
- B blocking factor used for work areas;
- IL number of physical index blocks per logical index block;
- IS index entry size;
- IB number of indexes/physical index block;
- RM maximum number of strings to be merged in one pass of Phase 2;
- EM maximum number of strings to be merged in Phase 3;
- BA base bin size;
- IX number of input buffers;
- OX number of final output buffers.

ICE121C ET=a CT=b BN=c X=d TO=e SN=f G=g

Explanation:

This message relates to Phase 1.

- ET elapsed time taken in centiseconds;
- CT CPU time in centiseconds;
- BN number of blocks handled;
- X number of EXCPs issued;
- TO number of tracks put out;
- SN number of strings produced;
- G number of records in the record storage area.

ICE122R ET=a CT=b BN=c X=d G RM=e PN=f BT=g TO=h

Explanation:

This message relates to Phase 2 (Reduction).

- ET elapsed time taken in centiseconds;
- CT CPU time in centiseconds;
- BN number of work data set blocks handled;
- X number of EXCPs issued;
- G number of records in the record storage area;
- RM maximum number of strings to be merged in one pass of Phase 2;
- PN highest partition number;
- BT number of tracks handled more than once.
- TO number of tracks put out;

```
                                    G
ICE123E  ET=a  CT=b  BN=c  X=d   EM =e  TO=f  BT=g
```

Explanation:

This message relates to Phase 3.

ET elapsed time taken in centiseconds;
CT CPU time in centiseconds;
BN number of work data set blocks handled;
X number of EXCPs issued;
G number of records in the record storage area;
EM maximum number of strings to be merged in Phase 3;
TO number of tracks put out;
BT number of tracks handled more than once.

```
ICE124P  ET=a  CT=b  PE=c  RP=d  CX=e  CO=f  C0=g  CR=h  G=i  WB=j
```

Explanation:

This message relates to Phase 2 (Partitioning).

ET elapsed time taken in centiseconds;
CT CPU time in centiseconds;
PE 'peerage': the number of logical strings obtained by logically rearranging the tracks of physical strings;
RP number of partitions;
CX number of exempt blocks;
CO number of overflow blocks;
C0 number of blocks in partition 0;
CR number of blocks to be handled by partition 0;
G number of records in the record storage area;
WB number of blocks written back to work storage.

```
ICE125O  CT=a  GP=b  SA=e  X=d
```

Explanation:

This message relates to work I/O, and is cumulative: it appears after each of Phases 1-3, and shows cumulative totals each time.

CT CPU time in centiseconds;
GP number of work I/O blocks;
SA number of standalone seeks;
X number of EXCPs issued.

Messages Produced by Using the DIAG Option

Diagnostic messages are obtained when you specify the DIAG option in the PARM field of the EXEC job control statement. This option is only available for tape techniques, a merge-only application, or when forcing a non-standard disk technique.

The DIAG option impairs program performance, and should be removed as soon as it is no longer needed.

The diagnostic messages are as follows:

ICE900I GENERATED CORE END ADDRxx	ICE926I IOB TBL ADDR xxxx
ICE901I INPUT BFR TBL ADDRxxxx	ICE927I I/P CCW ADDR xxxx
ICE902I OUTPUT BFR ADDR xxxx,xxxx	ICE940I GENERATED CORE END ADDR
ICE903I RSA TBL ADDR xxxx	ICE941I INPUT BFR TBL ADDR xxxx
ICE904I TREE ADR FROM xxxx TO xxxx	ICE942I OUTPUT BFR ADDR xxxx,xxxx
ICE905I MOVE RTN ADDR xxxx	ICE943I MOVE RTN ADDR xxxx
ICE906I DCB TBL ADDR xxxx	ICE944I ECB TBL ADDR xxxx
ICE907I O/P CCW ADDR xxxx	ICE945I I/P CCW ADDR xxxx
ICE908I OUTPUT IOB ADDR xxxx	ICE961I TECHNIQUE xxxx
ICE909I OPEN LIST ADDR xxxx	ICE962I NO/SIZE OF BFRS, PH x, x, xxxx
ICE920I GENERATED CORE END ADDR xxxx	ICE963I MAX.SYSGEN CORE xxxx
ICE921I INPUT BFR TBL ADDR xxxx	ICE964 CALC. CORE PH1=xxxx
ICE922I OUTPUT BFR ADDR xxxx,xxxx	ICE965I MERGE ORDER=xxxx
ICE9231 MOVE RTN ADDR xxxx	ICE988I ICEyyy LOC. AT xxxx[1]
ICE924I DCB TBL ADDR	ICE989I CLOCK - xx,xx,xx[2]
ICE925I O/P CCW ADDR xxxx	ICE990I NO OF STRINGS PROD BY PH1 xxxxxxx

[1]Appears frequently; provides the starting addresses of the program modules.

[2]Appears at the beginning of each phase (except Phase 0), and at the end of the program.

Dumps

There are two types of failure that can cause dumps:

- Sort-detected uncorrectable errors which give critical error messages.
- Sort program failures that are detected by the operating system.

NORMAL ABEND DUMPS

The default ERETINV|ERETJCL=ABEND|RC16, which was set when Sort was installed, can be overridden in a standard disk sort by the DEBUG control statement or by the PARM Field Option DIAG. To obtain a normal ABEND dump you must provide a SYSUDUMP or SYSABEND DD statement.

An invoked tape sort program, upon detecting an error, will give a return code of 16 unless DIAG is specified in the PARM Field Option, and a SYSUDUMP or SYSABEND DD statement is provided.

THE TRACE TABLE

In main storage the TRACE table is stored for Blockset in COMMON at COMTRACE (Register 13 + X'818') and for Peerage and Vale in ICECOMMA at CTRACE (Register 13 + X'814'). In a formatted dump it is eliminated from ICECOMMA dump and shown separately at the end of the FORMATTED DATA (see Figure 22). The TRACE table can hold up to 128 trace records, each one byte long. The table is divided into four fields:

FIELD 1	FIELD 2	FIELD 3	FIELD 4
4 bytes	32 bytes	32 bytes	60 bytes

The table is initialized with zeros.

How Events are Recorded

Four types of events are traced

Type	Blockset	Peerage and Vale
Type 1	Record level	Record level
Type 2	Block level	Block level
Type 3	Blockset level	Main loop level
Type 4	Mainly module level	Module level

The object is to keep a long trace of Type 4 events, a shorter trace of Types 3, and 2, and only a short trace of Type 1. For this reason record insertion is carried out as follows:

- A new event is inserted at the end of the field corresponding to the type number; Type 4 goes at the end of field 4, Type 3 at the end of field 3, etc.
- All records to the left of the new one move one byte to the left; the leftmost one (in Field 1) is thus lost.

Format of an Event Record

The Event record is one byte long, and consists of a description code. There are 255 codes (X'00' - X'FF').

EVENT-TABLE DESCRIPTION FOR BLOCKSET

TRACE	SEGMENT	MODULE	DESCRIPTION
		BLOCKSET	
1,01		ICECOBV	E15 CALL, COBOL MODE
1,01		ICEE15B	CALL E15
1,02		ICECOBV	E15 ANSWER, COBOL MODE
1,02		ICEE15B	RETURN FROM E15
1,03	C4COBIN	ICEIPUM	CALL E15
1,03		ICE15V	E15 CALL, SORTIN MODE
1,04	C4COBIN	ICEIPUM	RETURN FROM E15
1,04		ICE15V	E15 ANSWER, SORTIN MODE
1,07	C4LOAF35	ICEE35B	CALL E35
1,08	C4LOAF35	ICEE35B	RETURN FROM E35
1,09	C4LOAF35	ICEE35B	CALL E35
1,09	C5REXE35	ICE35VL	E35 CALL, SORTOUT MODE
1,0A	C4LOAF35	ICEE35B	RETURN FROM E35
1,0A	C5REXE35	ICE35VL	E35 ANSWER, SORTOUT MODE
1,0B	C5REXCOB	ICE35VL	E35 CALL, COBOL MODE
1,0C	C5REXCOB	ICE35VL	E35 ANSWER, COBOL MODE
		BLOCK LEVEL	
2,11	C4DXLBLD	ICEIPUB, KPUB, OPUB	DATA TRANSFER LIST BUILD
2,12	C4CHKIO	ICEIPUB, KPUB, OPUB	CHECK I/O TRANSFER
2,13	C4PUTVIR	ICEKPUV	PUT VIRTUIAL BLOCK RECORD
		BLOCKSET LEVEL	
3,21	C4PICKUT	ICEIPUM	PICK A BVD FOR OUTPUT
3,22	C4MAKKAD	ICEIPUM	MAKE KAD ENTRIES
3,23	C4CONVIM	ICEIPUB, KPUB, OPUB	CONVERT IRRR TO MBBCCHHR
3,24	C4CONVIM	ICEIPUB, KPUB, OPUB	CONVERT MBBCCHHR TO IRRR
3,26	C4PARBLK	ICEKPUA, OPUA	PARTITION INPUT BLOCKS
3,27	C4GETSPC	ICEKPUB	GET SORTWORK SPACE
3,28	C4KYBVDK	ICEKPUS, OPUT	KEY BVD MERGE
3,29	C4KEYIO	ICEKPUS, OPUT	KEY INPUT/OUTPUT
3,2A	C4COPIN	ICEKPUA, OPUA	COUNT RECORDS IN BUFFER
3,2B	C4COPOUT	ICEKPUA, OPUA	COUNT OUTPUT RECORDS IN RSA
3,2C	C4DUMPIN	ICEKPUA, OPUA	DUMP OUT INPUT BUFFERS
3,2D	C4HILOAD	ICEKPUA, OPUA	HIGH-TO-LOW SELECT LOAD ROUTINE
3,2E	C4ALTLST	ICEKPUA, OPUA	ALTERNATE LIST PREPARATION
3,2F	C4BNDSET	ICEKPUA, OPUA	SET BLOCK BOUNDS
3,30	C4MERGIN	ICEKPUA, OPUA	MERGE INPUT LISTS
3,31	C4PRUNE	ICEKPUA, OPUA	ADD A LIST TO SELECTION TREE
3,34	C4SETCMP	ICEKPUB, OPUB	SET COMPARE AND BRANCH MODES
3,36	C4STARK	ICEKPUB	START/STOP I/O
3,37	C4STARK	ICEKPUB	STOP SORTWORK INPUT
3,38	C4STARK	ICEKPUB	START SORTWORK OUTPUT
3,39	C4STARK	ICEKPUB	START KEY INPUT

EVENT-TABLE DESCRIPTION FOR BLOCKSET, Continued

BLOCKSET			
TRACE	SEGMENT	MODULE	DESCRIPTION
3,3A	C4STARK	ICEKPUB	START SORTWORK INPUT
3,3B	C4TRELL	ICEKPUB, OPUB	TREE LAD LIST RESTORE - HIGH TO LOW
3,3C	C4TRELL	ICEKPUB, OPUB	TREE LAD LIST RESTORE - LOW TO HIGH
3,3D	C4THIGH	ICEKPUS, OPUT	PREPARE TREE FOR HIGH TO LOW SELECT
3,3E	C4PIKBAK	ICEKPUS	PICK BOUNDARY VALUE FOR OUTBACK
3,40	C4KYCYLK	ICEKPUS	KEY CYLINDER
3,41	C4OUTBAK	ICEOPUT	WRITE BACK TO SORTWORK
3,42	C4FILLUT	ICEOPUA	FILL OUT OUTPUT AREA
3,43	C4LOADUT	ICEOPUA	LOAD SORTOUT BUFFERS
3,44	C4BACKUT	ICEOPUT	WRITE BACKOUT DATA
3,45	C4KYCYLO	ICEOPUT	KEY CYLINDER
3,46	C4KYBVDO	ICEOPUT	KEY BVD KAD MERGE
3,47		ICELIMI	TEST WRITEBACK LIMIT
3,48	C4TRKSET	ICEOPUA	TRACK SET ALLOCATION
3,49	C4RUNOUT	ICEOPUA	SELECT AN OUTPUT RUN LIST
3,4A	C4STOUT	ICEOPUB	STOP SORTWORK INPUT
3,4B	C4STOUT	ICEOPUB	START SORTWORK OUTPUT
3,4C	C4STOUT	ICEOPUB	START/STOP I/O
3,4D	C4STOUT	ICEOPUB	START SORTWORK INPUT
3,4F	C4ALCATA	ICEIPUK, KPUT	ALLOCATE AND INIT. I/O XFER AREA
3,4F	C4WAITIO	ICEIPUB, KPUB	PERFORM WAIT FUNCTION
3,50	C4OUTPUT	ICEOPUT	OUTPUT PASS
3,50		ICEIPVM	SORTIN TO RSA
3,51		ICEIPVM	BUILD SORTWORK BLOCKS
3,52		ICEIPVM	PREPARE TO WRITE TO SORTWORK
3,53		ICEIPVM	WRITE SORTWORK BLOCKS
3,54		ICEIPVM	SORTIN + E15 TO RSA
3,55		ICEIPVM	CALL ICECOBV. E15 TO RSA.
3,60		ICEKPVS	GET NEXT BLOCKSET KAD
3,61		ICEKPVS	EMPTY INPUT INTO RSA
3,62		ICEKPVS	WRITE SORTED OUTPUT, THEN EMPTY INPUT
3,63		ICEKPVS	WRITEBACK, THEN EMPTY INPUT
3,64		ICEKPVS	FLUSH OUTPUT
3,70	C5OUTPVT	ICEOPVT	WAIT ON BLOCKSET READ
3,71	C5OUTPVT	ICEOPVT	EMPTY INPUT BUFFERS
3,72	C5OUTPVT	ICEOPVT	ISSUE NEXT BLOCKSET READ
3,73	C5OUTPVT	ICEOPVT	MERGE DUMPED RECORDS INTO RSA RUN
3,74	C5OUTPVT	ICEOPVT	RESTORE SORTED OUTPUT

EVENT-TABLE DESCRIPTION FOR BLOCKSET, Continued

TRACE	SEGMENT	MODULE	DESCRIPTION
		BLOCKSET	
3,75	C5OUTPVT	ICEOPVT	PARTIAL DUMP MERGE DUMP RECS INTO INTO RSA RUN
3,76	C5OUTPVT	ICEOPVT	PARTIAL DUMP - RESTORE SORTED OUTPUT
3,77	C5UTBAKV	ICEOPVT	WRITEBACK TO SORTWORK
3,78	C5OUTPVT	ICEOPVT	CALL ICE35VL FOR COBOL RESTORE
3,79	C5OUTPVT	ICEOPVT	FLUSH RSA
3,7A	C5RESTOR	ICEOPVA	CALL ICE35VL FOR E35 INTERFACE SUPPORT
		MODULE LEVEL	
4,4E	C4STARIO	ICEIPUB, KPUB, OPUB	ISSUE EXCP
4,A0	C4KEYMRG	ICEKPUS, OPUT	UPDATE BLOCKSET KAD INDEX (WITH I/O)
4,A1	C4KEYPUT	ICEKUPS, OPUT	WRITE HIGH IN-CORE INDEX KADS TO DISK
4,A1	C4XAREA	ICEIPUT, KPUT	TRANSFER AREA INITIALIZATION
4,A3	C4EOVIN	ICEIPUM	HANDLE EOF ON SORTIN BEFORE SVC
4,A4	C4FINDIN	ICEIPUM	FIND SORTIN MBBCCHHR
4,A6	C4EOF	ICEIPUT	SET SORTOUT TRANSFER SPACE REQUIREMENTS
4,A7	C4DEBFIX	ICEIPUB, KPUB, OPUB	FIX DEVICE MODIFIERS BEFORE FIX SVC
4,A8		ICEE15B	RETURN FROM E15 WITH RC=8
4,A9	C4COBIN	ICEIPUM	RETURN FROM E15 WITH RC=8
4,AA	C4ADRESS	ICEIPUB, KPUT, OPUB	PUT REAL ADDRESSES IN IOBS, DXLS
4,AB	C4OSPACE	ICEKPUT	OPUT SPACE COMPUTATIONS
4,AC	C4KSPACE	ICEKPUT	KPUT TRANSFER SPACE COMPUTATIONS
4,AE	C4RELSE	ICEKPUT	RELEASE EXCESS DISK SPACE
4,AF	C4MAINFI	ICEKPUV	PREPARE MIAN STORAGE PRIMARY FILE
4,B0	C4LOAF35	ICEE35B	TRACE RC 8 RECEIVED
4,B1		ICEIPUT	INITIALIZE FOR INPUT PHASE
4,B2		ICEIPUT	REINITIALIZE AFTER SORTIN EOV
4,B4	C4INPUT	ICEIPUM	INPUT PASS
4,B5		ICEKPUT	INITIALIZE FOR PHASE 2 AND 3
4,B6		ICEKPUV	VIRTUAL BLOCK CONSTRUCTION
4,B7		ICEEXIK	EXIT FROM KEY PHASE BEFORE EXITTERM
4,B8	C4VOBLOK	ICEKPUV	VIRTUAL OUTPUT MERGE BLOCK CONSTRUCTION
4,B9	C4VIBLOK	ICEKPUV	VIRTUAL INTERUM MERGE BLOCK CONSTRUCTION
4,BB		ICEKPUS	KAD WRITEBACK MERGE (START OF MODULE)
4,BC		ICEKPUS	KAD WRITEBACK SORT
4,BD		ICEOPUT	HANDLE RECORDS AND INDEXES

EVENT-TABLE DESCRIPTION FOR BLOCKSET, Continued

TRACE	SEGMENT	MODULE	BLOCKSET	DESCRIPTION
			VLR MODULES	
4,C0	C5KEYCVL	ICEOPVT		BUILD BLOCKSET KADS FOR WRITEBACK CYL
4,C1	C5KADCRE	ICEOPVT		BUILD KAD FROM WBACK AREA IN-CORE INDEX
4,C2	C5KADCRE	ICEOPVT		BUILD CYL KAD FOR WRITEBACK AREA INDEX
4,C3	C5BAKUTV	ICEOPVA		TERMINATE WBACK LOAD DUE TO RSA END OF Q
4,D0		ICEIPVT		ICEIPVT ENTRY
4,D1		ICEIPVT		ALLOCATE ICEIPVT IOX
4,D2		ICEIPVT		BUILD ICEIPVT RSA
4,D3	C5CODVRT	ICEIPVT		CONVERT ICEIPVT CODE TO BINS
4,D4		ICEIPVM		IPVM ENTRY OR REENTRY
4,D5		ICEIPVM		PRIME SORTIN
4,D6		ICEIPVM		EOV ON SORTIN - E15
4,D7		ICEIPVM		EOV ON SORTIN - NO E15
4,D8		ICEIPVM		REINITIALIZE AFTER EOV - E15
4,D9		ICEIPVM		REINITIALIZE AFTER EOV - NO SORTIN
4,DA		ICEIPVM		EOF ON SORTIN - E15
4,DB		ICEIPVM		EOF ON SORTIN - NO E15
4,DC		ICEIPVM		RSA IS FLUSHED.
4,DF		ICEGENV		ICEGENV ENTRY
4,E0		ICEKPVT		ICEKPVT ENTRY
4,E1		ICEKPVT		BEGIN ICEKPVT ALLOCATION
4,E2		ICEKPVT		CREATE SORTED IN-CORE INDEX
4,E3	C5OPINIV	ICEKPVT		FREEMAIN AND REALLOCATE FOR ICEOPVT
4,E6		ICEKPVV		BUILD BLOCKSET KADS FROM SKAD KADS
4,E7		ICEKPVV		CALL KPVS TO SORT SMALL KADS
4,E8		ICEKPVV		BUILD BLOCKSET KADS USING KPVS OUTPUT
4,EA		ICEKPVV		BUILD BLOCKSET KADS FOR ICEOPVT
4,F0		ICEOPVT		ICEOPVT TRACE
4,F1	C5OUTPVT	ICEOPVT		RSA FLUSHED - SORTOUT
4,F2	C5OUTPVT	ICEOPVT		RSA FLUSHED - NO SORTOUT
4,F9		ICECOBV		RC 8 FROM E15, COBOL MODE
4,FA		ICE15V		RC 8 FROM E15, SORTIN CODE
4,FB	C5REXE35	ICE35VL		RC 8, SORTOUT MODE
4,FC	C5REXCOB	ICE35VL		RC 8, COBOL MODE

Event-Table Description for Peerage and Vale

---------- PEERAGE AND VALE ----------

TRACE	SEGMENT:	DESCRIPTION
1,01	CREMAIN:	TRACE E15 IN
1,01	VREMAIN:	TRACE E15 IN
1,01	VREMAINN:	TRACE E15 IN
1,02	CREMAIN:	TRACE E15 RETURN
1,02	VREMAIN:	TRACE E15 RETURN
1,02	VREMAINN:	TRACE E15 RETURN
1,03	CROOUT:	TRACE E35 ENTRY
1,03	VROOUT:	TRACE E35 ENTRY
1,04	CROOUT:	TRACE E35 RETURN
1,04	VROOUT:	TRACE E35 RETURN
1,05	VIMOUT:	TRACE E35 ENTRY
1,05	LIMOUT:	TRACE E35 ENTRY
1,05	LIVOUT:	TRACE E35 ENTRY
1,06	VIMOUT:	TRACE E35 RETURN
1,06	LIMOUT:	TRACE E35 RETURN
1,06	LIVOUT:	TRACE E35 RETURN
2,3E	WROUK:	WRITE TO WORK FILE
2,3F	VEDLAST:	WAIT FOR READ FROM WORK FILE
2,3F	WROUK:	WAIT FOR READ FROM WORK FILE
2,40	WROUK:	WRITE TO WORK FIELD
2,41	WROUK:	WAIT FOR WRITE TO WORK FILE
2,42	CREINIT:	ISSUE A READ
2,42	CREMAIN:	ISSUE A READ
2,42	VREMAIN:	ISSUE A READ
2,42	VREMAINN:	ISSUE A READ
2,42	VREMININ:	ISSUE A READ
2,42	VREMINIT:	ISSUE A READ
2,43	CREMAIN:	CHECK WAIT FOR READ
2,43	CREMAINN:	CHECK WAIT FOR READ
2,43	VREMAIN:	CHECK WAIT FOR READ
2,44	CROEND:	ISSUE A WRITE
2,44	VROEND:	ISSUE A WRITE
2,44	VROOUT:	ISSUE A WRITE
2,44	CROOUT:	ISSUE A WRITE
2,45	CROEND:	CHECK FOR WRITE
2,45	VROEND:	CHECK FOR WRITE
2,45	VROOUT:	CHECK FOR WRITE
2,45	CROOUT:	CHECK FOR WRITE
2,46	WORKFEOB:	DATA BLOCK WRITTEN ON AN INTERMEDIATE WORK AREA
2,48	REDSUBG:	EXCHANGE AND WRITE OUTPUT
2,4A	VEDLAST:	READ FROM WORK FILE
2,4A	VEDAHEAD:	START OF MERGE PASS
2,4B	VEDMAIN:	START OF MERGE PASS
2,4C	VEDMAIN:	START OF A SUBPASS
2,4F	REDSUBR:	WAIT FOR I/O
2,50	LIMRFILL:	TRACE EMTY/READ
2,52	LIMXWRT:	TRACE EXCHANGE / WRITE
2,53	LIMSUBR:	TRACE WAIT
2,54	LIVFLUSH:	TRACE SORTOUT WRITE
2,54	VIMMAIN:	TRACE SORTOUT WRITE
2,54	VIMOUT:	TRACE SORTOUT WRITE
2,54	LIMOUT:	TRACE SORTOUT WRITE
2,54	LIVOUT:	TRACE SORTOUT WRITE
2,54	VIMLAST:	TRACE SORTOUT WRITE
2,55	LIVFLUSH:	TRACE SORTOUT CHECK
2,55	VIMMAIN:	TRACE SORTOUT CHECK

---------- PEERAGE AND VALE ----------

TRACE	SEGMENT:	DESCRIPTION
2,55	VIMOUT:	TRACE SORTOUT CHECK
2,55	LIMOUT:	TRACE SORTOUT CHECK
2,55	LIVOUT:	TRACE SORTOUT CHECK
2,55	VIMLAST:	TRACE SORTOUT CHECK
3,B9	MOVERTBN:	PRIMING A NEW STRING
3,B9	EOSTRING:	END OF STRING PROCESSING
3,BA	CRESUBR:	BLOCK AN INDEX INTO AN INDEX OUTPUT BUFFER
3,BB	PARWRIT1:	WRITES A BLOCK OF WRITEVACK INDEXES DURING SORT OR FORWARD HASH PASS
3,BB	PARWRIT2:	WRITES A BLOCK OR WRITEBACK INDEXES DURING SORT OR FORWARD HASH PASS
3,BB	PARWRIT3:	WRITES A BLOCK OR WRITEBACK INDEXES DURING FINAL SORT
3,BC	PARWRTB0:	WRITES A BLOCK OF WRITEBACK INDEXES DURING SORT OR FORWARD HASH PASS
3,BC	PARWRTB1:	WRITES A BLOCK OR WRITEBACK INDEXES DURING SORT OR FORWARD HASH PASS
3,BD	PARREAD0:	READ A LOGICAL INPUT INDEX BLOCK
3,BD	PARREAD1:	READ A WRITEBACK BLOCK
3,BE	PARBLOCK:	BLOCKS A LOGICAL BLOCK OF INDEXES TO OUTPUT
3,BF	PARDBLK:	DEBLOCK INDEXES
3,C0	PARMRG:	MERGES THE CHAIN OF SORTED INDEXES
3,C1	PARSRT:	SORTS A DEBLOCKED LOGICAL INDEX BLOCK
3,C2	REDINIT:	TRACE SUBINDEX PRIME
3,C3	REDMAIN:	MAIN TRACK ENTRY
3,C4	REDREAD:	TRACE TRACK READ
3,C5	REDBACK:	GET WRITE BACK
3,C6	REDWRITE:	TRACE OUTPUT TRACK
3,C7	REDFLUSH:	WRITE LAST INDEX BUFFER AN WAIT ALL ACTIVE WORK I/O
3,CB	REDSUBR:	FIND RECORD TO START WRITE-BACK
3,CC	REDSUBR:	TRACE SEEK PROCESSING
3,CD	LIMMAIN:	TRACE MAIN ENTRY AND INITIATE READ
3,CE	LIMMAIN:	TRACE MAIN READ
3,CF	LIMBACK:	WRITEBACK
3,D0	LIMSUBR:	TRACE FINDBACK
3,D1	LIMSUBR:	TRACE INDEX GET
3,D2	LIMSUBR:	TRACE SEEKS
3,D4	BINALLC:	TRACE REPACK IN
3,D5	BINALLC:	TRACE REPACK OUT
3,D6	VREFLUSH:	ENTER USER CODE AT E17
3,D6	VREFLUSH:	TRACE EXIT IN ENTER USER EOD ROUTINE
3,D6	VREMAIN:	TRACE EXIT IN ENTER USER EOD ROUTINE
3,D6	VREMAINN:	TRACE EXIT IN ENTER USER EOD ROUTINE
3,D6	VREMININ:	TRACE EXIT IN ENTER USER EOD ROUTINE
3,D6	VREMININ:	ENTER USER CODE AT E16
3,D6	VREMINIT:	ENTER USER CODE AT E11
3,D6	VREMINIT:	ENTER USER CODE AT E16
3,D6	VROEND:	ENTER USER CODE AT E18
3,D6	VROEND:	ENTER USER CODE AT E17
3,D6	VROEND:	ENTER USER CODE AT E31
3,D6	VROEND:	ENTER USER CODE AT E37
3,D7	VREFLUSH:	BACK FROM USER CODE AT E17
3,D7	VREFLUSN:	TRACE EXIT OUT
3,D7	VREMAIN:	TRACE EXIT OUT
3,D7	VREMAINN:	TRACE EXIT OUT
3,D7	VREMININ:	TRACE EXIT OUT
3,D7	VREMININ:	BACK FROM USER CODE AT E16
3,D7	VREMINIT:	BACK FROM USER CODE AT E11
3,D7	VREMINIT:	BACK FROM USER CODE AT E16

| ---------- PEERAGE AND VALE ----------

TRACE	SEGMENT:	DESCRIPTION
3,D7	VROEND:	BACK FROM USER CODE AT E18
3,D7	VROEND:	BACK FROM USER CODE AT E17
3,D7	VROEND:	BACK FROM USER CODE AT E31
3,D7	VROEND:	BACK FROM USER CODE AT E37
4,DE	SORTWOUT:	TO AVOID EXIT THRU MANEXIT AT I/O TASK
		PUT A MARK IN THE TRACE TABLE
4,DF		ICECRE ENTRY
4,DF		ICEVRE ENTRY
4,E0	CREINIT:	TRACE PRIME RSA
4,E0	VREMININ:	TRACE PRIME RSA
4,E0	VREMINIT:	TRACE PRIME RSA
4,E1	CREINIT:	INITIALIZE WORK FILE ROUTINE
4,E1	VREMININ:	INITIALIZE WORK FILE ROUTINE
4,E1	VREMINIT:	INITIALIZE WORK FILE ROUTINE
4,E2	CROEND:	RESTORE TREE REGISTER AND GET WORK AREA ADDR
4,E2	VROEND:	RESTORE TREE REGISTER AND GET WORK AREA ADDR
4,E3	CREFLUSH:	FLUSH RECORDS REMAINING IN RSA TO WORK FILES
4,E3	VREFLUSH:	ENTER USER EOD ROUTINE, IF DEFINED
4,E3	VREFLUSN:	ENTER USER EOD ROUTINE, IF DEFINED
4,E4		ICEPAR ENTRY
4,E5	PARSORT0:	READ THE INPUT INDEX FILE
4,E6	PARHAS0:	FINISHES ORDERING OF INDEXES
4,E7	PARHAS1:	PROCESSES INDEXES PRODUCED BY PARHS0
4,E8	PARSORT1:	ORDER THE INDEXES FOR REDUCTION PHASE
4,E9		ICEPAR EXIT
4,EA	PARTERM:	FLUSH REMAINING PROCESSED INDEXES
4,EB	VEDPROLG:	ICEVED ENTRY
4,EB		ICERED
4,EC	EXITPH13:	ICEVED EXIT
4,EC		ICERED EXIT
4,ED		ICELIM ENTRY
4,ED		ICELIV ENTRY
4,EE		ICELIM EXIT
4,EE		ICELIV EXIT
4,EE		ICEVIM EXIT
4,EF		ICECRE EXIT
4,EF		ICEVRE EXIT
4,EF		ICEVRN EXIT
4,F0		ICEDEF ENTRY
4,F1		ICEDEC ENTRY
4,F2		ICEDEG ENTRY
4,F3		ICEDEV EXIT

FORCING A SPECIALLY FORMATTED DUMP

This option is only available for Peerage and Vale.

The default ERETJCL|ERETINV=ABEND|RC16, which was set when Sort was installed, can be overridden in a standard disk sort by the DEBUG control statement.

To obtain a specially formatted dump for a sort, the CTRx=value must be specified in the DEBUG statement. This first prints a SNAP dump (corresponding to a normal SYSUDUMP dump), followed by formatted information as shown in Figure 22.

The formatted part of the dump is derived from the source code, which is written in the PL/S language. As shown in Figure 23 this is not difficult to interpret, even without a knowledge of PL/S. However, if you would like to know more about the language you can find much information, including how to read source code, in the publication Guide to PL/S, GC28-6794.

Example:

```
   DEBUG  ABEND,CTR2=1
```

(1) SYSTEM DUMP
SNAP dump corresponding to a normal SYSUDUMP dump.

(2) FORMATTED DATA

2.1 Save areas
The standard save areas used by different levels of the program.

2.2 ABEND code
A fullword with the format X'xxsssuuu', where
xx is the standard ABEND code prefix,
sss is the system completion code at program failure (or zeros), and
uuu is the user completion code at uncorrectable error (or zeros).
This code is equal to zero for definition errors, and equal to the message number for other errors (for example, '046' would represent message ICE046A).

2.3 A fullword containing the address of the instruction at which failure occurred.

2.4 Register contents when program failure occurred: 16 fullwords giving the register contents in the order 0-15.

2.5 Contents of ICECOMMA (sort variables) formatted when program failure occurred, with offsets from Register 13, comments, labels, and definitions.

2.6 Trace of important events, in the form

x yyy

where x identifies the part of the program

yyy identifies the segment of code entered.

x can be one of the following codes:

DEF - definition (ICEDEF)
C - creation (ICECRE, ICEVRE, ICEVRN)
P - partitioning (ICEPAR)
R - reduction (ICERED, ICEVED)
E - elimination (ICELIM, ICEVIM, ICELIV, ICEVIP)
A - appendage (for PCI, channel end, or end-of-extent)

The first event listed is the most recent*; the last is the first that occurred (normally DEF ENTRY).

*If one of the most recent events listed concerns an exit, the probable cause of program failure is a programming error in the exit routine.

Figure 22. Contents of a Specially Formatted Dump

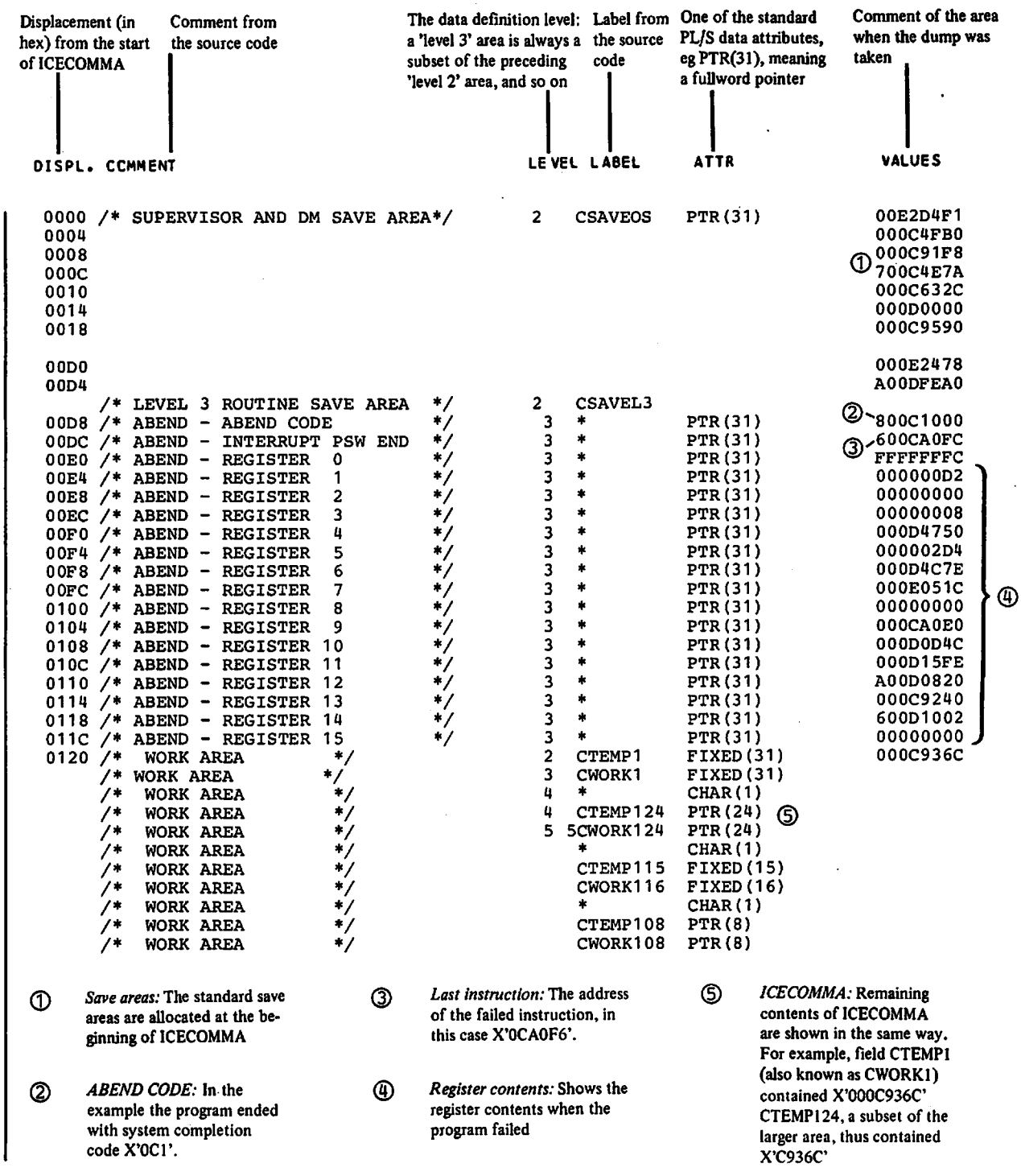


Figure 23. Interpreting a Formatted Dump

Finding an Object Module in a Storage Dump

As shown in Figure 24, the name of each module appears at the beginning of the module in the storage dump.

```
  LPA/JPA MCDULE    ICEMCN
  0B7CC0   C9C3C5D4 E6D54040 D9C5D3 40  *ICEMCN  REL 5.0 PTF 00  CCPYRIGH*
  0B7020   E340C9C2 D440C3D6 D9D74E 40  *T IBM CCRP. 1973......C.........*
  0B7040   18214100 03004510 CC120 ACA  *.............................L....*
  0B7060   45F0CC3C 000B7C6C 00000 JCCC  *.C..........ICECEF  ..K.L....0A.*
  0B7080   D507D418 BC90477C CC60' 95E8  *N.M........YL......CJ...N.M.....*
  0B70A0   C0FE41A0 BD1050A0 D44C 95E8  *.........M..YL......MM..........*
  0B70C0   47F0CC92 41CCCCC4 CA3C .411C  *.C..........L..CJ.....L..CJ.....*
  0B70E0   D39858F0 D14405EF C20 3D320  *L..CJ...K.L.JH.TCJ.....MM.......*
  0B7100   005B58FC C160C5EF D2( )3C320  *...CJ...K.L.JH..............0..*
  0B7120   45F0F016 000B712C 00' 000000  *.00.........ICERCC  ....M...J...*
  0B7140   50A0D134 18F15C2C FC .C00A06  *..J..l..C.....M.............C.*
  0B716C   C74FC203 C68CF9C8 9' 3ECFCCC  *..K.L.C...0...J...M....C....A..C*
  0B718J   C16247F0 C18447F0 C 1C847F0  *A..CA..CAH.CA8.0B..0B..0A2.0....*
  0B71AC   1CC4C2J0 100AE0C1 F )FFFD1CC  *..K.......J.......A...J..CB.....*
  0B71C0   5010D1CC 47F0C196 I )60C10CA  *..J..0A.D..........A......0A.N.*
  0B71E0   100AC4F4 4770C1C2  C2032C1C  *..D4..ABK..........A..CB....CA.*
  0B720C   1EF088F0 CC1654FC  C4EC5EFC  *.C.C...CC..CC..CC..CC...C...B...*
  0B7220   201041F0 000447F0 C1FC1222  *...C...CA.....B..0........B..0..*
  0B7240   96402000 45EGC5C8 47F0C226  *. ....EF.CB.....................  *
  0B7260   20C48F0F 20044780 C2BA5500  *     .....B...DD..B.P
  0B7280   [illegible] 41101014 , 41F0F[illegible]      .....00..[illegible]
  0B7[illegible]   58DD0C0[illegible]      .....[illegible]
                   003[illegible]
```

Figure 24. The Start of a Module in a Dump

Each module starts with a header consisting of the eight-byte module name followed by the release and PTF number. Some modules have a copyright notice visible after the header.

The module entry point comes after this preliminary material.

With the Blockset technique each load module consists of many object modules. Details are given in Section 3 under 'Phase Structures'. Each object module starts with a header as for the other techniques.

Origin of Program Messages

This subsection shows which modules issue each message. It also tells you how to locate a given message in the source code. This can be useful if, for example, you encounter an error in Phase 0, and can thus not get a dump.

BLOCKSET TECHNIQUE

To find a Blockset message in the code, you should look in the cross reference for labels with the following format:

MSG0xxy

where xx is the message number and y is a letter of the alphabet. The letter A means this is the first use of the message in this module, B means it is the second, and so on. For example, MSG046B would be the second occurrence of message ICE046A.

Message	Module	Message	Module	Message	Module
000	EXIN	046	IPUM	071	COBU
010	INIB		IPVM		COBV
013	INIB		KPUB		E15B
015	COBV		KPUT		E35B
	IPVM		OPUA		OPVT
	15V		KPVT		15V
	35VL	047	IPUT		35VL
017	INIC		IPVT	073	COBV
018	INIB	052	EXIO		IPVM
019	EXIN		OXOV		15V
021	INIB	054	EXIO		35VL
023	INIB		OXOV	074	EXIN
024	INIB	055	EXIO	080	EXIO
025	EXIO		OXOV	081	INIC
	OXOV	056	INIB	083	DYNA
026	EXIO	059	INID	085	EXIO
	OXOV	061	IPUB		OXOV
027	INIC		IPVB	088	EXIN
039	INIO		KPUB	089	EXIN
	IPUT		KPVB	092	EXIN
	IPVT		OPUB	093	EXIN
	KPUT		OPVB	094	INIC
	KPVT	063	INIB	096	MESS
	SUBS		MAN	098	OXOV
042	MAN	067	EXIN	099	IPUM
043	IPUM				IPVB
	INIB				
	IPVT				
	MAN				

ALL OTHER TECHNIQUES

To find a Peerage or Vale message in the code, you should look in the cross reference for labels with the following format:

DIAGxxy

where xx is the message number and y is a letter of the alphabet. The letter A means this is the first use of the message in this module, B means it is the second, and so on. For example, DIAG46B would be the second occurrence of message ICE046A.

Exceptions to the above rule occur in modules ICEDEC and ICEDED, which also use tables for issuing messages. There you should look for labels:

SERRxxy
or
SNERRxx

where xx is the message number, and y is the letter L or P.
other Message Cross-Reference Tables

Technique->	Disk						Tape						Merge
Message V	PEER	VALE	CRCX fix	CRCX var	BALN fix	BALN var	OSCL fix	OSCL var	POLY fix	POLY var	BALN fix	BALN var	-only
000	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
001	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
002	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
003	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
004	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
005	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF
	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
006	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
007	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
008	DEC	DEC	DEC[1]	DEC[1]	DEC[1]	DEC[1]	DEC[1]	DEC[1]	DEC[1]	DEC[1]	DEC[1]	DEC[1]	DEC
010	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]
012	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
013	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
014	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
015 no E	-	VRE[2]	-	RDS	-	RDG	-	RDS	-	RDG	-	RDG	-
Exits	-	VRE[2]	-	RDT	-	RDE	-	RDT	-	RDE	-	RDE	-
016	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
017	DED	DED	DED	DED	DED	DED	DED	DED	DED	DED	DED	DED	DED
	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG
018	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
020	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
021	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]
022	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]
023	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]
024	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]
025 ph1	CRO	VRO	-	-	-	-	-	-	-	-	-	-	-
ph3	LIM[3]	VIM	-	-	-	-	-	-	-	-	-	-	-
027	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]	DEC[8]
028	-	-	RCH[4]	RCH[4]	RCH[4]	RCH[4]	RCH[4]	RCH[4]	RCH[4]	RCH[4]	RCH[4]	RCH[4]	RCH
029	DEC	DEC	DEC[4]	DEC[4]	DEC[4]	DEC[4]	DEC[4]	DEC[4]	DEC[4]	DEC[4]	DEC[4]	DEC[4]	DEC
030	DEC	DEC	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH
031	-	-	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH
032	DEC[8]	DEC[8]	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH
033	DEC[8]	DEC[8]	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH
034	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
035	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
036	-	-	BGB	BGB	RCK	RCK	RCS	RCS	RCS	RCS	RCS	RCS	-
037	-	-	BGB	BGB	RCK	RCK	RCS	RCS	RCS	RCS	RCS	RCS	-
038	-	-	RCJ	RCJ	RCJ	RCJ	RCS	RCS	RCS	RCS	RCS	RCS	-
039 ph0	DEV	DEV	BGB	BGB	RCK	RCK	RCS	RCS	RCS	RCS	RCS	RCS	RCL
ph1	CRE	VRE[2]	-	-	-	-	-	-	-	-	-	-	-
ph2	PAR	-	-	-	-	-	-	-	-	-	-	-	-
ph3	LIM[3]	VIP	-	-	-	-	-	-	-	-	-	-	-
all	MON	MON	-	-	-	-	-	-	-	-	-	-	-
040	-	-	RCI[5]	RCI[5]	RCI[5]	RCI[5]	RCI	RCI	RCI	RCI	RCI	RCI	-
041	-	-	RCJ	RCJ	RCJ	RCJ	-	-	-	-	-	-	-
042	DEG[5]	DEG[5]	DEG[7]	DEG[7]	DEG[7]	DEG[7]	DEG[7]	DEG[7]	DEG[7]	DEG[7]	DEG[7]	DEG[7]	DEG[7]

[1]Or RCD (if the error is in a MODS statement parameter)
[2]Or VRN
[3]Or LIV
[4]Or RCH
[5]RCJ if the error is that insufficient tracks were allocated (rather than an insufficient number of devices)
[7]Or RCI, DEF, or DEC
[8]Or DED

Technique->	Disk						Tape						Merge
Message V	PEER	VALE	CRCX fix	CRCX var	BALN fix	BALN var	OSCL fix	OSCL var	POLY fix	POLY var	BALN fix	BALN var	-only
043	DEC[8]	DEC[8]	DED[9]	DED[9]	DED[9]	DED[9]	DED[9]	DED[9]	DED[9]	DED[9]	DED[9]	DED[9]	DED[9]
044 ph0	DEC[14]	DEC[14]	-	-	-	-	-	-	-	-	-	-	-
ph1	-	-	9GN	9GN	AGI	AGI	AGN	AGN	AGA	AGA	AGA	AGA	-
ph2	-	-	9GN	9GN	AGJ	AGJ	AGN	AGN	AGG	AGG	AGG	AGG	-
ph3	-	-	AGK	AGK	AGK	AGK	APK	APK	APK	APK	APK	APK	APF
045	-	-	-	-	RPC	RPC	-	-	RPC	RPC	RPC	RPC	-
046 all	MON	MON	-	-	-	-	-	-	-	-	-	-	-
ph1	CRE	VRE[3]	8ON	8ON	ROK	ROK	RON	RON	ROJ	ROJ	ROI	ROI	-
ph2	RED	-	8ON	8ON	-	-	RON	RON	ROS	ROS	ROR	ROR	-
047 ph1	CRE	VRE[2]	8ON	8ON	RPC	RPC	RON	RON	RPC	RPC	RPC	RPC	-
	CRO	VRO	-	-	-	-	-	-	-	-	-	-	-
ph2	-	-	8PM	8PM	RPF	RPF	RPM	RPM	RPF	RPF	RPF	RPF	-
ph3	-	-	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
048	-	-	RDR	RDS	RDD	RDE	RDR	RDS	RDD	RDE	RDD	RDE	-
049	-	-	-	-	RPC	RPC	-	-	RPC	RPC	RPC	RPC	-
050	-	-	8PM	8PM	RPF	RPF	RPM	RPM	RPF	RPF	RPF	RPF	-
051	-	VED	-	-	-	-	-	-	-	-	ROR	ROR	-
052	-	-	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
053	MON	MON	ROQ[10]	ROQ[10]	ROQ[10]	ROQ[10]	ROQ[10]	ROQ[10]	ROQ[10]	ROQ[10]	ROQ[10]	ROQ[10]	-
054	LIM[3]	VIM	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
	CRO	VRO	-	-	-	-	-	-	-	-	-	-	-
055	LIM[3]	VIM	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
	CRO	VRO	-	-	-	-	-	-	-	-	-	-	-
056	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
	DED	DED	RCI	RCI	RCI	RCI	RCI	RCI	RCI	RCI	RCI	RCI	RCI
057	-	-	-	-	-	-	RCI	RCI	RCI	RCI	RCI	RCI	-
058	-	-	-	-	-	-	RCI	RCI	RCI	RCI	RCI	RCI	-
059	DEC	DEC	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]
060	-	-	RC4	RC4	RC4	RC4	-	-	-	-	-	-	-
061 ph0	-	-	RCB	RCB	RCB	RCB	RCB	RCB	RCB	RCB	RCB	RCB	RCB
ph1 read	MON[11]	MON[11]	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV
write	MON[11]	MON[11]	8PA	8PA	RPB	RPB	RPA	RPA	RPA	RPA	RPA	RPA	-
ph2 rd b[15]	-	-	-	-	-	-	RGB[13]	RGB[13]	RGB[13]	RGB[13]	RGB[13]	RGB[13]	-
rd f[15]	MON	MON	8GB	8GB	RGC[13]	RGC[13]	-	-	RGL[13]	RGL[13]	RGL[13]	RGL[13]	-
write	MON	MON	8PA	8PA	RPE	RPE	RPA	RPA	RPD	RPD	RPD	RPD	-
ph3 rd b[15]	-	-	-	-	-	-	RGD[13]	RGD[13]	RGD[13]	RGD[13]	RGD[13]	RGD[13]	-
rd f[15]	MON[11]	MON[11]	8GC[13]	8GC[13]	RGE[13]	RGE[13]	-	-	RGM[13]	RGL[13]	RGM[13]	RGM[13]	RCV
write	MON[11]	MON[11]	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV
062	DEC[12]	DEC[12]	RCO	RCO	RCO	RCO	RCO	RCO	RCO	RCO	RCO	RCO	RCO
063 ph0	DEF[14]	DEF[14]	RCM[14]	RCM[14]	RCM[14]	RCM[14]	RCM[14]	RCM[14]	RCM[14]	RCM[14]	RCM[14]	RCM[14]	RCM[14]
ph1	CRE	VRE[2]	APC	APC	APC	APC	APC	APC	APC	APC	APC	APC	-
	CRO	VRO	-	-	-	-	-	-	-	-	-	-	-
ph2	-	-	-	-	APJ	APJ	-	-	APJ	APJ	APJ	APJ	-
ph3	-	-	AGH	AGH	AGH	AGH	AGH	AGH	AGH	AGH	AGH	AGH	AGF
064	-	-	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV	RCV
065	DEC	DEC	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH

[2]Or VRN
[3]Or LIV
[6]Or DEF or DEC
[8]Or DED
[9]Or RCI
[10]ROP if multiple compare routine used
[11]Or VRE, VRN, VRO, XCP, or OBS
[12]Or DEF, DEC, or DED
[13]Or the corresponding assignment module
[14]Or DEC or DED
[15]rd b means read backward (tape), rd f means read forward (tape), read (disk)

Technique->	D i s k						T a p e						Merge
Message V	PEER	VALE	CRCX fix	CRCX var	BALN fix	BALN var	OSCL fix	OSCL var	POLY fix	POLY var	BALN fix	BALN var	-only
066 ph1	-	-	8ON	8ON	ROK	ROK	RON	RON	ROJ	ROJ	ROI	ROI	-
ph2	-	-	8ON	8ON	-	-	RON	RON	ROS	ROS	ROR	ROR	-
067	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF
068	-	-	-	-	-	-	-	-	-	-	-	-	ROQ[9]
069 'single'	-	-	ROB	ROD	ROB	ROD	ROB	ROD	ROF	ROH	ROB	ROD	BALN)
'multiple'	-	-	ROA	ROC	ROA	ROC	ROA	ROC	ROE	ROG	ROA	ROC	(as
070	-	-	-	-	RCK	RCK	-	-	-	-	-	-	-
071 ph1	CRO	VRE[2]	RDR	RDS	RDD	RDE	RDR	RDS	RDD	RDE	RDD	RDE	-
	-	VRO	-	-	-	-	-	-	-	-	-	-	-
ph2	-	VEE	RBW	RBX	RBJ	RBK	RBW	RBX	RBJ	RBK	RBJ	RBK	-
ph3	LIM[18]	VIM	RBM	RBO[16]	RBM	RBO[16]	RBM	RBO[16]	RBM	RBO[16]	RBM	RBO[16]	RBM[17]
072	DED	DED	-	-	-	-	-	-	-	-	-	-	-
	-	DEC	-	-	-	-	-	-	-	-	-	-	-
073 no E	-	VRE[2]	-	RDT	-	RDG	-	RDT	-	RDG	-	RDG	-
Exits	-	VRE[2]	-	RDS	-	RDE	-	RDS	-	RDE	-	RDE	-
074	DEC[8]	DEC[8]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]	DEC[9]
075[19]	ERR	ERR	-	-	-	-	-	-	-	-	-	-	-
076	ERR	ERR	RGV	RGV	RGV	RGV	RGV	RGV	RGV	RGV	RGV	RGV	RGV
077	ERR	ERR	RPV	RPV	RPV	RPV	RPV	RPV	RPV	RPV	RPV	RPV	RPV
078 ph0[19]	ERR	ERR	RCI	RCI	RCI	RCI	RCI	RCI	RCI	RCI	RCI	RCI	RCI
ph1	-	-	APC	APC	APC	APC	APC	APC	APC	APC	APC	APC	-
ph3 in	-	-	-	-	-	-	-	-	-	-	-	-	AGF[9]
out	ERR	ERR	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
079 ph0	ERR	ERR	-	-	-	-	-	-	-	-	-	-	-
ph1	-	-	RCT	RCT	RCT	RCT	RCT	RCT	RCT	RCT	RCT	RCT	-
ph3 in[20]	-	-	-	-	-	-	-	-	-	-	-	-	RGF
ph3 out[20]	-	-	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
080	CRO	VRO	-	-	-	-	-	-	-	-	-	-	-
081	DEV	DEV	-	-	-	-	-	-	-	-	-	-	-
082	-	VED	-	-	-	-	-	-	-	-	-	-	-
083	DEG	DEG	-	-	-	-	-	-	-	-	-	-	-
084	DEV	DEV	-	-	-	-	-	-	-	-	-	-	-
085 ph0	DEC	DEC	-	-	-	-	-	-	-	-	-	-	-
ph1	CRO	VRO	-	-	-	-	-	-	-	-	-	-	-
ph3	LIM[3]	VIM	-	-	-	-	-	-	-	-	-	-	-
all	MON	MON	-	-	-	-	-	-	-	-	-	-	-
087	MON	MON	-	-	-	-	-	-	-	-	-	-	-
088	DEV	DEV	-	-	-	-	-	-	-	-	-	-	-
089	DEV	DEV	-	-	-	-	-	-	-	-	-	-	-
090	DEC	DEC	-	-	-	-	-	-	-	-	-	-	-
091	MON	MON	-	-	-	-	-	-	-	-	-	-	-
092	DEV	DEV	-	-	-	-	-	-	-	-	-	-	-
093	DEV	DEV	-	-	-	-	-	-	-	-	-	-	-
099	XCP	XCP	-	-	-	-	-	-	-	-	-	-	-

[2]Or VRN
[3]Or LIV
[8]Or DED
[9]Or RCI
[16]8BO if records are spanned
[17]RBO if variable non-spanned records, 8BO if spanned
[18]Can be issued by any of a large number of modules, including all those which generate control blocks, open, close, and block output. Look in the dump at the address given in the message.
[19]With a conventional technique can be RCI, DED or RCZ
[20]RCT if a sort terminates unsuccessfully

Cross-Reference Tables

CPI-PPI CROSS-REFERENCE TABLE

The following table contains the names of CPI and PPI fields together with their displacements; the last three letters of the names of modules that reference them; and the manner in which the modules are related to them. A dash is used to separate the module name from the related code. The codes are as follows:

U The information in the field is used by the module.

M The information in the field is created or modified by the module. It may also be used.

O Non-specific relationship exists between the module and the field. Either or both of the other two codes may, or may not, be applicable.

Field	Displacement	Addressed by Modules						
CPIADDCF	(EB)	RCI-U	RCM-M	RCN-U	RC1-U	RC2-U		
CPIAQTB	(133)	RCM-M	RC1-U					
CPIATP1E	(C7)	RCI-U	RCM-M	RC1-U				
CPIATP3E	(CD)	RCI-U	RCL-U	RCM-M	RC1-U			
CPIAUENT	(112)	RCH-M	RC1-U					
CPIBINSZ	(C2)	BGA-U	BGB-U	RCN-M	RCR-U	RCS-U	RC1-U	
CPIBUF1	(E5)	BGA-M	BGB-M	RC1-U				
CPIBUF23	(AE)	BGA-M	BGB-M	RCL-M	RC1-U			
CPIDCBIN	(FA)	BGB-U	RCI-M	RCL-U	RCS-U	RC1-U	8CI-M	
CPIDCBOU	(FC)	BGB-U	RCI-M	RCL-U	RCS-U	RC1-U	8CI-M	
CPIDDSRT	(F6)	RCI-U	RCM-M	RC1-U				
CPIDEV	(102)	BGB-U	RCI-M	RC1-U				
CPIDEVCD	(132)	RCI-M	RC1-U					
CPIFFF	(A4)	RC1-U	RC2-M					
CPIFILSZ	(D3)	RCM-M	RCS-U	RC1-U	8CI-M			
CPIIPBLK	(A2)	BGB-U	RCI-M	RCS-U	RC1-U			
CPIJOBNM	(11F)	RCM-M	RCZ-U	RC1-U				
CPILAB03	(BC)	BGA-M	BGB-U	RCI-M	RCL-U	RCS-U	RC1-U	
CPILAB07	(BE)	BGA-M	BGB-M	RCL-M	RCS-M	RC1-U	RC2-M	
CPILAB09	(9A)	BGA-M	BGB-M	RCI-M	RCL-M	RCS-U	RC1-U	
CPILINK	(E8)	RCH-M	RCM-U	RCO-M	RCZ-U	RC1-U		
CPIMODEX	(CA)	BGA-U	BGB-U	RCH-U	RCI-M	RCL-U	RCM-M	RCS-U RC1-U
CPIMRGAL	(E7)	BGA-M	BGB-M	RCI-M	RCL-U	RCM-M	RC1-U	
CPIMRGMX	(E6)	BGA-M	BGB-M	RCL-M	RC1-U			
CPIMSGDD	(116)	RCB-U	RCM-M	RCO-M				
CPIMXCOL	(130)	RC1-U	RC2-M					
CPINUMCF	(48)	RCI-U	RCM-M	RCN-U	RC1-U	RC2-U		
CPINWKU	(9E)	BGB-U	RCI-M	RCL-U	RCN-U	RCS-U	RC1-U	
CPIOPBLK		RCI-M	RC1-U					
CPIPBUFF	(A6)	RC1-U	RC2-M					
CPIPCF01	(4A)	RCI-O	RCM-M	RCN-O	RC1-U	RC2-O		
CPIPGCSZ	(C0)	BGB-U	RCL-U	RCS-U	RC1-U	RC2-M		
CPIP1ASZ	(A0)	RC1-U						
CPIP1GC	(DC)	BGA-M	BGB-M					
CPIP1RSZ	(DC)	BGB-U	RCI-M	RCS-U	RC1-U			
CPIP2GC	(DF)	BGA-M	BGB-M					
CPIP2RSZ	(DF)	BGB-U	RCS-U	RC1-U				
CPIP3ASZ	(AA)	RCB-U	RCM-M	RCZ-U	RC1-U			
CPIP3GC	(E2)	BGA-M	BGB-M					

Field	Displacement	Addressed by Modules							
CPIP3RSZ	(E2)	BGB-U	RCI-M	RCL-M	RCS-U	RC1-U			
CPIRCDL1	(B2)	BGA-U	BGB-U	RCI-M	RCL-U	RCM-M	RCR-U	RCS-U	RC1-U
		8CI-U							
CPIRCDL2	(B4)	BGA-U	BGB-U	RCI-M	RCL-U	RCM-M	RCN-U	RCR-U	RCS-U
CPIRCDL3	(B6)	BGA-U	BGB-U	RCI-M	RCL-U	RCM-M	RCS-U	8CI-U	
CPIRCDL4	(B8)	RCI-M	RCM-M	RCN-U					
CPIRCDL5	(BA)	BGB-U	RCI-M	RCM-M	RCN-U	RCR-U	RCS-U		
CPIRECFM	(11E)	RCI-M	RC1-U						
CPISESDS		8CI-M							
CPISIRG	(D9)	BGA-M	RC1-U						
CPISKPRD	(C4)	RCM-M	RC1-U						
CPISKSDS		8CI-M							
CPISMAX		RCI-U	RCM-M						
CPISORCE	(EE)	RCH-M	RC1-U						
CPISRTBL	(B0)	BGA-M	BGB-M	RCI-M	RCS-U	RC1-U			
CPISRTG	(D6)	BGA-M	BGB-M	RC1-U					
CPISTPNM	(127)	RCM-M	RCZ-U						
CPISUENT	(115)	RCH-M	RC1-U	RC2-U					
CPISVSAM		RCI-U	8CI-U						
CPISVSIN		BGB-U	RCI-U	RCL-U	RCS-U	8CI-M			
CPISVSOP		BGB-U	RCI-U	RCL-U	RCR-U	RCS-U	8CI-M		
CPISW1	(92)	BGA-M	BGB-M	RCD-M	RCH-M	RCI-M	RCL-M	RCM-M	RCN-M
		RCP-U	RCS-U	RCZ-U	RC1-U	RC2-U	8CI-M		
CPISW14		BGB-U	RCI-M	RCN-U					
CPISW17		RCN-M							
CPISW2	(12F)	BGB-U	RCI-U	RCL-U	RCM-M	RCO-U	RCR-U	RCS-U	RCZ-U
		RC1-U	8CI-M						
CPITAVLC	(D0)	BGA-M	BGB-M	RCL-U	RCM-M	RCS-U	RC1-U		
CPIXCAP	(F8)	BGA-M	RC1-U						
PPIADDCF	(418)	AOM-U	RCZ-U	RC1-M					
PPIADSSC	(324)	ABF-U	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	ABQ-U	ABR-U
		ABS-U	AGA-U	AGC-U	AGD-U	AGE-U	AGF-U	AGG-U	AGI-U
		AGJ-U	AGK-U	AGL-U	AGM-U	AGN-U	AOA-U	AOB-U	AOC-U
		AOD-U	AOE-U	AOF-U	AOG-U	AOH-U	AOM-U	APA-U	APB-U
		APD-U	APE-U	APF-U	APG-U	APH-U	API-U	APK-U	APL-U
		RBJ-U	RBK-U	RBM-U	RBO-U	RBW-U	RBX-U	RCT-U	RCV-M
		RC6-U	RC7-U	RC8-U	RDD-U	RDE-U	RDG-U	RDR-U	RDS-U
		RDT-U	RDU-U	RGB-U	RGL-U	RGM-U	RGV-U	ROI-U	ROJ-U
		ROK-U	RON-U	ROP-U	ROQ-U	ROR-U	ROS-U	ROT-U	RPA-U
		RPB-U	RPC-U	RPD-U	RPE-U	RPF-U	RPG-U	RPM-U	RPV-U
		8BO-U	8ON-U	9BN-U	9BO-U	9GB-U	9GC-U	9GN-U	
PPIALG	(328)	AOI-U	AOJ-U	AOK-U	AON-U	AOR-U	AOS-U	AOT-U	RCT-O
		RC6-O	RC7-M	RC8-M	RC9-O	RDR-U	RDS-U	RGB-U	RGC-U
		RGL-U	ROW-U	ROY-U	RPA-U	RPB-U	RPD-U	RPE-U	RPG-U
		8GB-U	8PA-U	9ON-U					
PPIAMA	(3A9)	ABF-U	ABS-U	AGA-U	AGI-U	AGN-U	AOA-U	AOB-U	AOC-U
		AOD-U	AOE-U	AOF-U	AOG-U	AOH-U	APA-U	APB-U	APC-U
		APG-U	APL-U	RC9-O	9GN-U	9PA-U			
PPIAMB	(3A8)	ABR-U	AGB-U	AGC-U	AGG-U	AGJ-U	AGL-U	AGM-U	APD-U
		APE-U	APH-U	APJ-U	APL-U	9GB-U			
PPIAMC	(3A8)	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	ABQ-U	AGD-U	AGE-U
		AGF-U	AGH-U	AGK-U	APF-U	API-U	APK-U	9BN-U	9BO-U
		9GC-U							
PPIAPGC	(432)	RCB-U	RC1-M	RC6-U	RC7-U	RC8-U			
PPIATP1E	(54)	ADD-U	ADE-U	ADR-U	ADS-U	ADU-U	AOQ-U	APC-U	APF-U
		RC1-M	RC6-U						
PPIATP3E	(2C4)	ABM-U	ABO-U	APF-U	RC1-M	RC8-U	RPG-U	9BO-U	
PPIAXERT	(29C)	AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH-U
		AOL-U	AOM-U	AOP-U	AOQ-U	RC1-M	ROW-U	ROY-U	RPC-U
		8ON-U							

Field	Displacement	Addressed by Modules							
PPIBDSVA	(2E8)	ABC-U	ABF-U	ABH-U	ABJ-U	ABQ-M	ABR-M	ABS-M	ABT-U
		ABU-U	ABV-U	ABW-U	ABX-U	ABZ-U	ADE-U	ADG-U	ADH-M
		ADI-M	ADL-M	ADS-M	ADT-M	ADX-M	AGA-U	AGB-U	AGC-U
		AGD-U	AGE-M	AGG-U	AGH-U	AGL-U	AGM-U	AGN-U	AOI-M
		AOJ-M	AON-M	AOQ-U	APA-U	APC-M	API-M	APJ-U	APK-U
		RBA-M	RBC-U	RBE-M	RBG-U	RBH-U	RBI-U	RBJ-U	RBK-U
		RBM-U	RBP-U	RBT-U	RBU-U	RBV-U	RBW-U	RBX-U	RBZ-U
		RCT-U	RDC-U	RDD-U	RDE-M	RDG-M	RDI-U	RDL-U	RDQ-U
		RDR-U	RDS-M	RDT-M	RGB-U	RGL-U	ROK-M	RON-U	ROS-U
		RPC-U	RPF-U	RPM-U	8ON-U	9BN-U	9BO-U	9GB-U	9GC-U
		9ON-M							
PPIBINSZ	(260)	ABF-U	ADE-U	ADG-U	ADP-U	ADQ-U	ADS-U	ADT-U	APC-U
		APG-U	APL-U	RCK-U	RC1-M	RDB-U	RDC-U	RDD-U	RDP-U
		RDQ-U	RDR-U	RDS-U	RDT-U				
PPIBLK	(340)	ABA-U	ABB-U	ABC-U	ABE-U	ABL-U	ABM-U	ABN-U	ABO-U
		ABP-U	ABY-U	ABZ-U	ROA-U	ROB-U	ROC-U	ROD-U	ROE-U
		ROF-U	ROG-U	ROH-U	ROP-U	ROQ-U	RPA-U	RPB-U	RPG-U
		8PA-U	9BN-U	9BO-U					
PPIBLK2	(370)	ABG-U	ABH-U	ABI-U	ABJ-U	ABK-U	ABT-U	ABU-U	ABV-U
		ABW-U	ABX-U	RGB-U	RGC-U	RGL-U	ROP-U	ROQ-U	RPA-U
		RPD-U	RPE-U	8GB-U	8PA-U				
PPIBPTRK	(2AA)	AGC-U	AGE-U	RCJ-M	ROK-U	RPB-U	RPE-U	8CK-M	8GB-U
		9PA-U							
PPIBUF1	(5C)	APB-U	APG-M	APL-U	RCJ-U	RCK-M	RC1-M	8CK-M	
PPIBUF23	(274)	APH-U	API-U	APL-U	RCK-M	RC1-M	8CK-U		
PPICHKAD	(424)	AGH-U	APC-U	APF-U	APJ-U	RC9-M	RON-U	ROS-U	ROT-U
PPICONV	(380)	ABF-U	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	ABQ-U	ABR-U
		ABS-U	AGA-U	AGB-U	AGC-U	AGD-U	AGE-U	AGG-U	AGI-U
		AGJ-U	AGK-U	AGL-U	AGM-U	AGN-U	AOA-U	AOB-U	AOC-U
		AOD-U	AOE-U	AOF-U	AOG-U	AOH-U	APA-U	APB-U	APD-U
		APE-U	APF-U	APG-U	APH-U	API-U	APK-U	APL-U	RC6-M
		RC7-U	RC8-M	RC9-U	ROI-U	ROJ-U	ROK-U	RON-U	8ON-U
		9BN-U	9BO-U	9GB-U	9GC-U	9GN-U	9PA-U		
PPICOUNT	(240)	ABG-M	ABH-M	ABJ-M	RBG-M	RBH-M	RBI-M	RBJ-M	RBK-M
		RBL-M	RBM-M	RBN-M	RBO-M	RBP-M	RDB-M	RDC-M	RDD-M
		RDE-M	RDG-M	RDP-M	RDQ-M	RDR-M	RDS-M	RDT-M	ROA-U
		ROB-U	ROC-U	ROD-U	ROE-U	ROF-U	ROG-U	ROH-U	ROI-U
		ROJ-U	ROK-U	RON-U	ROR-U	RPC-M	RPE-U	RPF-M	RPG-U
		RPM-M	8BN-M	8BO-M	8ON-U	8PM-M			
PPICPTRK	(430)	RCJ-M	8CK-M	8PA-U	9GN-U				
PPIDCBIN	(428)	AGA-U	AGI-U	AGN-U	APF-U	RCK-U	RC1-M	9GN-U	
PPIDCBOU	(42A)	AGK-U	APF-U	APK-U	RCK-U	RC1-M			
PPIDDSRT	(41C)	AGA-U	AGG-U	AGH-U	AGI-U	AGJ-U	AGK-U	AGN-U	APC-U
		APF-U	APJ-U	APK-U	RC1-M	RC4-U	RPG-U	9GN-U	
PPIDD0L1	(298)	AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH-U
		AOP-U	AOQ-M	RC1-M					
PPIDEB	(330)	ABC-U	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADH-U	ADI-U
		ADJ-U	ADP-U	ADQ-U	ADR-U	ADS-U	ADT-U	ADU-U	ADX-U
		AGA-U	AGI-U	AGN-U	RBA-U	RBB-U	RBC-U	RBE-U	RBL-U
		RBM-U	RBN-U	RBO-U	RBP-U	RBY-U	RBZ-U	RGD-U	RGE-U
		RGM-U	ROA-U	ROB-U	ROC-U	ROD-U	ROE-U	ROF-U	ROG-U
		ROH-U	ROI-U	ROJ-U	ROK-U	RON-U	ROW-U	ROX-U	ROY-U
		ROZ-U	8BN-U	8BO-U	8GC-U	8ON-U	9DJ-U	9GN-U	
PPIDEB2	(360)	ADL-U	RGB-U	RGC-U	RGL-U	8GB-U			
PPIDELCT	(244)	RBJ-M	RBK-M	RBW-M	RBX-M	RDE-M	RDR-M	RDS-M	ROA-U
		ROB-U	ROC-U	ROD-U	ROE-U	ROF-U	ROG-U	ROH-U	ROI-U
		ROJ-U	ROK-U	RON-U	8ON-U				

Field	Displacement	Addressed by Modules							
PPIDEPHQ	(280)	ABA-U	ABT-U	ABU-U	AEV-U	ABW-U	ABX-U	ABY-U	ABZ-U
		AGA-M	AGG-U	AGI-U	AGN-M	AOI-M	AOJ-M	AOK-M	AON-M
		AOR-M	AOS-M	APA-U	ADP-U	RBA-U	RBT-U	RBU-U	RBV-U
		RBW-U	RBX-U	RBY-U	REZ-U	RGB-U	ROI-U	ROJ-U	ROK-U
PPIDEPHO	(280)	RON-M	ROR-M	ROS-M	ROT-M	RPA-U	RPB-M	RPD-U	RPE-M
		8GB-U	8ON-M	8PA-M	9ON-M				
PPIDEV	(240)	APB-M	RCJ-M	RCK-U	RC1-M	8CK-U			
PPIDEVCD	(452)	AGC-U	AGE-U	APB-U	APE-U	RC1-M	8CK-U	9GB-U	9GC-U
		9PA-U							
PPIDIRAD	(118)	AGE-O	RGC-U	ROK-M	ROT-M	RPB-U	RPE-U		
PPIDOOBA	(2E4)	RBL-M	RBM-M	RBN-M	RBO-M	RBP-M	ROP-M	ROQ-U	RPG-U
		8BN-U	8BO-U	9BN-M	9BO-M				
PPIDOUO	(2B4)	API-M	RBM-M	RBO-M	ROP-U	ROQ-U	8BO-M		
PPIDPTRK	(42E)	RCJ-M	ROK-U	ROT-U					
PPIDSKED	(98)	AGB-O	AO1-M	RCJ-O	RC4-M	ROK-U	8CK-O	8ON-M	8PM-O
PPIENDAR	(1A8)	AGE-O	AGH-O	AOK-O	AOT-O	AO1-O	APJ-O	RCJ-O	ROK-M
		RPB-O	RPC-O	RPE-O	RPF-O	RPM-O	8CK-U	8ON-O	8PM-O
		9GC-O	9ON-O						
PPIEOF	(388)	AGA-U	AGI-U	AGN-U	APF-U	RCI-O	RCV-M	RC7-M	RC8-M
		RGV-U	9GN-U						
PPIFCF01	(77)	AOL-U							
PPIFFF	(64)	AOM-U	RC1-M						
PPIFILSZ	(25C)	APB-U	RCJ-U	RCK-M	RC1-M	RPC-M	RPF-M	RPG-U	RPM-M
		8CK-U	8PM-M						
PPIFSEON	(23B)	RPC-U	RPM-U	8PM-U					
PPIFTTAB	(43C)	APB-M	RC1-M	RPB-U	RPE-U				
PPIGETMN	(30C)	ADE-U	ADG-U	ADS-U	ADT-U	APC-U	APG-M	APH-M	API-M
		APL-M	RCT-U	RCV-U	RPA-U	RPB-U	RPD-U	RPE-U	RPG-U
		8PA-U	9DJ-U						
PPIGETSZ	(310)	APG-M	APH-M	API-M	APL-M	RCI-U	RCV-U	RPA-U	RPB-U
		RPD-U	RPE-U	RPG-U	8PA-U				
PPIINSCT	(248)	RDE-M	RDR-M	RDS-M	ROA-U	ROB-U	ROC-U	ROD-U	ROE-U
		ROF-U	ROG-U	ROH-U	ROI-U	ROJ-U	ROK-U	RON-U	8ON-U
PPIINT	(378)	AOW-U	AOX-U	AOY-U	AOZ-U	RON-U	8ON-U		
PPIIPBLK	(58)	RCK-U	RC1-M						
PPIJOBNM	(430)	RCV-U	RCZ-M	RC1-M					
PPILABO1	(C8)	AGB-M	AGC-M	AGD-M	AGE-M	AGM-M	AOI-O	AO5-M	ROT-O
		RPB-M	RPE-M						
PPILAB02	(2D4)	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADP-U	ADq-U	ADR-U
		ADS-U	ADT-U	AGA-U	AGB-U	AGC-U	AGD-M	AGE-M	AGF-U
		AGI-U	AGL-M	AGM-M	AGN-U	APF-U	APG-M	APH-M	API-M
		APL-M	RCT-U	RGD-M	RGE-M	RGM-M	8GC-M	9GC-M	9GN-U
PPILAB03	(2AC)	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADH-U	ADL-U	ADP-U
		ADQ-U	ADR-U	ADS-U	ADT-U	AGA-U	AGB-M	AGC-M	AGD-M
		AGE-M	AGF-U	AGI-U	AGL-M	AGM-M	AGN-U	APF-U	APG-M
		APH-M	API-M	APL-M	RCK-U	RCT-U	RC1-M	RGB-U	RGC-U
		RGD-U	RGE-U	RGL-U	RGM-U	8GC-U	9GC-U	9GN-U	
PPILAB04	(2D8)	ABA-U	ABB-U	ABC-U	ABE-U	ABG-U	AGH-U	ABI-U	ABJ-U
		ABK-U	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	ABT-U	ABU-U
		ABV-U	ABW-U	ABX-U	ABY-U	ABZ-U	AGK-U	APA-U	APB-U
		APD-U	APE-U	APF-U	APG-M	APH-M	API-M	APK-U	APL-M
		RPA-U	RPB-U	RPD-U	RPE-U	RPG-U	RPV-U	8PA-U	9BN-U
		9BO-U	9PA-U						
PPILAB05	(2DC)	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	APA-U	APB-U	APD-U
		APE-U	APG-M	APH-M	API-M	APL-M	RPA-M	RPB-M	RPD-M
		RPE-M	RPG-O	8PA-M	9BN-U	9BO-U	9PA-U		
PPILAB06	(2E0)	AGA-U	AGI-U	AGK-U	AGN-U	APF-U	APG-M	API-M	APK-U
		APL-M	RPG-U	9GN-U					

Field	Displacement	Addressed by Modules							
PPILAB07	(2B0)	ABA-U	ABB-U	ABC-U	ABE-U	ABG-U	ABH-U	ABI-U	ABJ-U
		ABK-U	ABT-U	ABU-U	ABV-U	ABW-U	ABX-U	ABY-U	ABZ-U
		ADL-U	AGB-U	AGK-U	APA-U	APB-U	APD-U	APE-U	APF-U
		APG-M	APH-M	API-M	APK-U	APL-M	RBA-U	RBE-U	RBI-U
		RBK-U	RBV-U	RBX-U	RCJ-U	RCK-M	RC1-M	RGB-U	RPA-M
		RPB-M	RPD-M	RPE-M	RPG-M	8GB-U	8PA-M	9GB-U	9PA-U
PPILAB08	(2E4)	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADP-U	ADQ-U	ADR-U
		ADS-U	ADT-U	APG-M	APL-M	RDP-U	RDQ-U	RDR-U	RDS-U
		RDT-U							
PPILAB09	(2B4)	APD-U	API-U	APL-U	RCK-M	RC1-M			
PPILAB10	(308)	AGB-U	APL-M	9GB-U					
PPILEXFD	(2A4)	AOB-U	AOD-U	AOF-U	AOH-U	AOM-M	AOQ-U		
PPILEXFF	(2A5)	ABF-U	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADP-U	ADQ-U
		ADR-U	ADS-U	ADT-U	AOB-U	AOD-U	AOF-U	AOH-U	AOM-M
		AOQ-U	RBA-U	RBE-U					
PPILINK	(23C)	RCZ-U	RC1-M	RC6-U	RC7-U	RC8-U			
PPIMODEX	(238)	ABQ-U	ABR-U	ABS-U	ADD-U	ADE-U	ADG-U	ADR-U	ADS-U
		ADT-U	AGA-U	AGG-U	AGI-U	AGJ-U	AGK-U	AGN-U	AOB-U
		AOD-U	AOF-U	AOH-U	APA-U	APB-U	APC-U	APF-U	APG-U
		APH-U	API-U	APK-U	APL-U	RCH-M	RCK-U	RC1-M	RC6-U
		RC7-U	RC8-U	RON-U	RPC-U	RPF-U	RPG-U	RPM-U	8CK-U
		8PM-U	9GN-U	9PA-U					
PPIMOVEQ	(458)	ABS-M	RDC-U	RDD-U	RDQ-U	RDR-U			
PPIMRGAL	(294)	ADU-M	AOI-U	AOJ-M	AOQ-U	AOR-U	AO2-U	APH-U	RCK-M
		RC1-M	ROI-U	ROR-M	ROT-U				
PPIMRGMX	(292)	ADH-U	ADI-U	ADJ-M	ADL-U	ADR-U	ADS-U	ADU-M	ADX-U
		AGA-U	AGB-U	AGD-U	AGE-M	AGF-U	AGI-U	AGM-U	AGN-U
		AOI-U	AOJ-U	AON-U	AOQ-U	AOR-U	AOS-M	AOT-M	AO2-U
		APF-U	APH-U	API-U	AOK-U	APL-U	RCJ-U	RCK-M	RCL-M
		RCT-U	RC1-M	RDL-U	RGB-U	RGC-U	RGL-U	ROI-M	ROK-U
		RON-M	ROP-U	ROQ-U	ROR-M	ROS-M	ROT-M	RPC-U	8GB-U
		8GC-U	8ON-M	9DJ-M	9GB-U	9GC-U	9ON-U		
PPIMRGOP	(296)	ADL-U	AGC-U	AGJ-U	AOQ-U	AOR-M	APH-M	ROK-M	ROT-M
		9ON-M							
PPIMXCOL	(450)	RC1-M	RDE-U	RDG-U	RDS-U	RDT-U	ROP-U	ROQ-U	
PPINDSKA	(2A8)	AGE-U	AGI-U	AGJ-U	AGK-U	AOK-U	AOT-U	AO1-U	APB-U
		APE-U	APH-U	RCJ-U	RCK-U	RCT-U	RC1-M	RC4-U	ROK-U
		ROT-U	RPB-U	RPC-U	RPE-U	RPF-U	RPG-U	8CK-U	8ON-U
		8PM-U	9GN-U	9ON-U					
PPINET	(338)	AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH-U
		AOP-U	AOQ-U	AOW-U	AOX-U	AOY-U	AOZ-U	RDB-U	RDC-U
		RDD-U	RDE-U	RDG-U	RDH-U	RDI-U	RDJ-U	RDP-U	RDQ-U
		RDR-U	RDS-U	RDT-U	RDU-U	RDX-U	RGD-U	RGE-U	RGF-U
		RGM-U	ROJ-U	ROW-U	ROX-U	ROY-U	ROZ-U	RPA-U	RPB-U
		RPG-U	8DJ-U	8GC-U	8PA-U				
PPINETAR	(438)	ADH-M	ADI-M	ADJ-M	ADL-M	ADU-M	ADX-M	AGE-M	RDL-U
		RGC-U	ROP-U	ROQ-U	ROT-U	9DJ-M			
PPINETM	(368)	AOP-U	AOQ-U	RBG-U	RBH-U	RBI-U	RBJ-U	RBK-U	RBT-U
		RBU-U	RBV-U	RBW-U	RBX-U	RDL-U	RGB-U	RGC-U	RGL-U
		8GB-U							
PPINUMCF	(70)	AOL-U	AOM-U	AO5-U	RCZ-U	RC1-M			
PPIODOM	(120)	AON-O	RON-O						
PPIOPBAD	(454)	ABG-M	ABH-M	ABJ-M	ABT-M	ABU-M	ABW-M	RBG-M	RBH-M
		RBI-M	RBJ-M	RBK-M	RBT-M	RBU-M	RBV-M	RBW-M	RBX-M
		ROP-U	ROQ-U						
PPIOPBLK	(272)	AGK-U	APF-U	APK-U	RC1-M				
PPIOPEN	(3B0)	AGA-U	AGB-U	AGC-U	AGD-U	AGE-U	AGG-U	AGH-U	AGI-U
		AGJ-U	AGK-U	AGL-U	AGM-U	AGN-U	APA-U	APB-U	APC-U
		APD-U	APE-U	APJ-U	APK-U	RCT-O	RPA-U	9GB-U	9GC-U
		9GN-U	9PA-U						

Field	Displacement	Addressed by Modules							
PPIPBUFF	(68)	AOM-U	RC1-M						
PPIPCF01	(72)	AOL-O	AOM-O						
PPIPDWA	(48)	ADH-O	ADI-O	ADJ-O	ADU-O	ADX-O	RDI-O	ROK-M	RGP-U
		RPQ-U	9DJ-O						
PPIPGCSZ	(60)	RCB-U	RCK-U	RC1-M					
PPIPSVA	(88)	ABQ-M	ABR-M	ABS-M	ADJ-M	AGB-M	AGL-M	APC-M	API-M
		RGB-O	RGD-O	RGF-U	RGL-O	RGM-O	8GB-O	8GC-O	9DJ-M
PPIP1ASZ	(4C)	RC1-M							
PPIP1GC	(48)	APG-U	APL-U	RCK-M	RC1-M				
PPIP2GC	(2B8)	APH-U	RCK-M	RC1-M					
PPIP3ASZ	(2C0)	RCV-U	RC1-M						
PPIP3GC	(2BC)	API-U	RCK-M	RC1-M					
PPIRCDCT	(24C)	ADH-U	ADI-I	AGD-U	AGH-U	AGM-U	APK-U	RPC-M	RPF-M
		RPG-U	RPM-M	8PM-M	9GC-U				
PPIRCDL1	(288)	ADB-U	ADP-U	AGA-U	AGI-U	AGN-U	APF-U	APG-U	API-U
		APL-U	RCK-U	RC1-M	RDE-U	RDS-U	9GN-U		
PPIRCLD2	(28A)	ABB-U	ABC-U	ABG-U	ABH-U	ABJ-U	ABR-U	ABS-U	ABT-U
		ABU-U	ABW-U	ABY-U	ABZ-U	ADH-U	ADL-U	ADX-U	AGD-U
		AGE-U	AGM-U	APB-U	APE-U	RBB-U	RBC-U	RBG-U	RBH-U
		RBJ-U	RBT-U	RBU-U	RBW-U	RBY-U	RBZ-U	RCJ-U	RCK-U
		RC6-U	RDE-U	RDG-U	RDH-U	RDL-U	RDS-U	RDT-U	RDX-U
		RGB-U	RGC-U	RGE-U	RGL-U	RGM-U	8CK-U	8GB-U	8GC-U
PPIRCDL3	(28C)	ABL-U	ABQ-U	AGK-U	APF-U	API-U	APK-U	RCK-U	RC8-U
PPIRCDL4	(28E)	RCJ-U	ROI-U	ROJ-U	ROK-U	RON-U	ROR-U	RPE-U	8ON-U
PPIRCDL5	(290)	APG-U	RCJ-U	RCK-U	ROS-U	8CK-U			
PPIRCV		AGA-U	AGB-U	AGD-U	AGE-M	AGF-U	AGH-U	AGI-U	AGK-U
		AGM-M	AGN-U	AOM-U	APC-U	APF-U	APJ-U	APK-U	RCV-M
		RC9-U	RGB-M	RGC-M	RGD-M	RGE-M	RGL-M	RGM-M	ROA-U
		ROB-U	ROC-U	ROD-U	ROE-U	ROF-U	ROG-U	ROH-U	ROI-U
		ROJ-U	ROK-U	RON-U	ROR-U	ROS-U	RPA-M	RPB-M	RPD-M
		RPE-M	RPG-U	8GB-M	8GC-M	8PA-M	8PM-U	9GN-U	9PA-U
PPIRD	(358)	AGB-U	AGC-U	AGD-U	AGE-U	AGL-U	AGM-U	RBG-U	RBH-U
		RBI-U	RBJ-U	RBK-U	RBT-U	RBU-U	RBV-U	RBW-U	RBX-U
		RCV-U	RDH-U	RDI-U	RDX-U	RON-U	ROR-U	ROS-U	ROT-U
		8ON-U	9BG-U	9GC-U					
PPIRECFM	(276)	AGK-U	APF-U	APK-U	RC1-M				
PPIRMA	(390)	AGA-U	AGI-U	AGN-U	AOJ-U	RDD-U	RDE-U	RDG-U	RDR-U
		RDS-U	RDT-U	RGV-U	ROI-U	ROJ-U	ROK-U	RON-U	RPC-U
		8ON-U	9GN-U						
PPIRMB	(398)	RBJ-U	RBK-U	RBW-U	RBX-U	ROP-U	ROQ-U	ROR-U	ROS-U
		RPE-U	RPF-U	RPM-U	8PM-U				
PPIRMC	(398)	RBM-U	RBO-U	RDU-U	RGV-U	ROP-U	ROQ-U	RPG-U	RPV-U
		8BO-U							
PPIRPLDP	(45C)	AGA-M	AGI-M	AGK-M	AGN-M	APF-M	APK-M	RBN-U	RBO-U
		RGV-U	RPG-U	RPV-U	9GN-M				
PPISBLCT	(2FC)	ADH-U	ADI-U	ADX-U	AGB-M	AGC-M	AGD-M	AGE-M	AGL-M
		AGM-M	RBG-U	RBH-U	RBI-U	RBJ-U	RBK-U	RBT-U	RBU-U
		RBV-U	RBW-U	RBX-U	RDH-U	RDI-U	RDL-U	RDX-U	RGB-U
		RGC-U	RGD-U	RGE-U	RGL-U	RGM-U	8GB-U	8GC-U	9GB-M
		9GC-M							
PPISEQCT	(250)	AGE-O	AGK-O	AOR-M	AOT-O	APH-O	ROI-M	ROK-O	ROR-M
		ROT-O							
PPISESDS		AGK-U	APF-U	APK-U					
PPISIRG	(264)	RC1-M	RPA-U						
PPISKPRD	(50)	APC-U	RC1-M						
PPISKSDS		AGK-U	APF-U	APK-U					
PPISLIB	(31C)	RCV-U	RCZ-M	RC6-U	RC7-U	RC8-U	RC9-U		
PPISORCE	(314)	RCZ-M	RC1-M	RC9-U					

Field	Displacement	Addressed by Modules							
PPISPGN1	(2D0)	ABQ-M	ABR-M	ABS-M	ABT-M	ABU-M	ABV-M	ABW-M	ABX-M
		ADH-M	ADI-M	ADJ-M	ADL-M	ADU-M	ADX-M	AGA-M	AGB-M
		AGC-M	AGD-M	AGE-M	AGG-M	AGI-M	AGJ-M	AGK-M	AGL-M
		AGM-M	AGN-M	AOA-M	AOB-M	AOC-M	AOD-M	AOE-M	AOF-M
		AOG-M	AOH-M	AON-M	AOQ-M	APA-M	APB-M	APD-M	APE-M
		APF-M	APG-M	APH-M	API-M	APK-M	APL-M	RCT-U	9DJ-M
		9GB-M	9GC-M	9GN-M	9ON-M	9PA-M			
PPISRALG	(264)	AGE-U	AGI-M	AGJ-M	AGK-M	RGC-U	RGE-U	ROT-M	RPB-U
		RPE-M	8GB-U	8GC-U	8PA-U	9GN-M			
PPISRTEL	(270)	ABB-U	ABC-U	ABG-U	ABH-U	ABJ-U	ABT-U	ABU-U	ABW-U
		ABY-U	ABZ-U	ADH-U	ADL-U	AGD-U	AGE-U	AGM-U	APB-U
		APE-U	APG-U	RBB-U	RBC-U	RBG-U	RBH-U	RBJ-U	RBT-U
		RBU-U	RBW-U	RBY-U	RBZ-U	RCJ-U	RCK-M	RC1-M	RGB-U
		RGD-U	RGL-U	RGM-U	ROI-U	ROJ-U	ROK-U	RON-U	ROR-U
		ROS-U	RPB-U	RPE-U	8GB-U	8GC-U	8ON-U	9GC-U	
PPISRTG	(26C)	AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH-U
		APG-M	APL-M	RCJ-U	RCK-M	RC1-M	ROI-U	ROJ-U	ROK-U
		ROW-U	ROX-U	ROY-U	ROZ-U				
PPISTAR	(120)	AGI-O	AGJ-O	AGK-O	AOK-O	AOT-O	AO1-O	APB-O	APE-O
		RCJ-O	8CK-O	9GN-O	9ON-O				
PPISTDCB	(2F8)	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	ADB-U	ADC-U	ADD-U
		ADE-U	ADG-U	ADJ-U	ADP-U	ADQ-U	ADR-U	ADS-U	ADT-U
		AGA-M	AGB-U	AGD-U	AGE-U	AGF-U	AGG-M	AGH-U	AGI-M
		AGJ-M	AGK-M	AGL-U	AGM-U	AGN-M	APA-U	APB-U	APC-U
		APD-U	APE-U	APF-M	APJ-U	APK-M	RCT-U	RGB-U	RGC-U
		RGD-U	RGL-U	RGM-U	RON-U	ROS-U	RPA-U	RPB-U	RPC-U
		RPD-U	RPE-U	RPF-U	RPG-U	RPM-U	8GB-U	8GC-U	8PA-U
		8PM-U	9BN-U	9BO-U	9DJ-U	9GB-U	9GC-U	9GN-M	
PPISTIOB	(300)	AGA-M	AGC-U	AGE-U	AGI-M	AGJ-M	AGK-M	APB-U	APD-U
		APE-U	RGC-U	RGE-U	RPB-U	RPE-U	8GC-U	9GC-U	9GN-M
		9PA-U							
PPISTPNM	(448)	RCV-U	RCZ-M						
PPISVARE	(0)	AGD-M	AGM-M						
PPISVSIN		AGA-U	AGF-U	AGI-U	AGN-U	APC-U	APF-U	APG-U	APL-U
		RCT-U	RC6-U	RC8-U	RGF-U	9GN-U			
PPISVSOP		AGK-U	APF-U	APK-U	REN-U	RBO-U	RCT-U	RC8-U	RPG-U
PPISW1	(230)	ABA-U	ABB-U	ABC-U	ABE-U	ABF-U	ABG-U	ABH-U	ABI-U
		ABJ-U	ABK-U	ABL-U	AEM-U	ABN-U	ABO-U	ABP-U	ABQ-U
		ABR-U	ABS-U	ABT-U	ABU-U	ABV-U	ABW-U	ABX-U	ABY-U
		ABZ-U	ADH-U	ADI-U	ADJ-M	ADL-U	ADP-M	ADQ-M	ADR-M
		ADS-M	ADT-M	ADU-M	AGA-U	AGB-M	AGC-M	AGD-M	AGE-M
		AGG-U	AGH-U	AGI-U	AGJ-U	AGK-U	AGI-M	AGM-M	AGN-U
		AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH_U
		AOL-U	AON-U	AOP-U	AOQ-U	AOR-M	AOS-M	AOT-U	AO1-U
		AO2-U	APA-U	APB-M	APC-U	APD-U	APE-U	APF-U	APG-M
		APH-M	API-M	APJ-U	APK-U	APL-M	RBM-M	RBO-U	RCB-U
		RCH-M	RCJ-U	RCK-M	RCT-U	RCV-M	RCZ-M	RC1-M	RC4-U
		RC6-U	RC7-U	RC8-M	RC9-U	RDD-M	RDE-M	RDL-U	RDP-M
		RDQ-M	RDR-M	RDS-M	RDT-M	RDU-U	RGB-M	RGC-M	RGD-M
		RGE-M	RGF-U	RGL-M	RGM-M	RGV-U	ROI-M	ROJ-M	ROK-U
		RON-M	ROP-U	ROQ-U	ROR-M	ROS-M	ROT-M	ROW-U	ROY-U
		RPA-M	RPB-M	RPC-M	RPD-M	RPE-M	RPF-M	RPG-U	RPM-M
		8BO-U	8CK-U	9GB-M	8GC-M	8ON-M	8PA-M	8PM-M	9BN-U
		9BO-U	9DJ-M	9GB-M	9GC-M	9GN-U	9PA-U		
PPISW14		AGC-U	AGE-U	AGI-U	AGJ-U	AGK-U	APB-U	APE-U	RGC-U
		RGE-U	RPB-U	RPE-U	8GB-U	8GC-U	9GB-U	9GC-U	9GN-U
		9PA-U							

Field	Displacement	Addressed by Modules							
PPISW2	(23B)	ABS-U	AGA-U	AGF-U	AGI-U	AGK-U	AGN-U	AOM-U	AO5-U
		APC-U	APF-U	APG-U	APK-U	APL-U	RBI-U	RBM-U	RBN-U
		RBO-U	RBP-U	RCB-U	RCK-U	RCT-U	RC2-M	RC1-M	RC6-U
		RC8-U	RDB-U	RDC-U	RDD-U	RDE-U	RDG-U	RDI-U	RDP-U
		RDQ-U	RDR-U	RDS-U	RDT-U	RGF-U	RPC-U	RPG-U	RPM-U
		8BN-U	8BO-U	8PM-U	9GN-U				
PPITAVLC	(2C8)	APG-U	APL-U	RCK-M	RC1-M				
PPITPCYL	(42C)	AGC-U	AGE-U	AOK-U	APB-U	APE-U	RCJ-M	ROT-U	8CK-M
		9ON-U							
PPITPPT	(F4)	AOR-M	AO2-U	APK-U	RC4-M	ROI-M			
PPITPTBL	(F8)	AGA-O	AGB-O	AGD-O	AGG-O	AGH-O	AGI-O	AGM-O	AGN-O
		AOI-O	AOJ-O	AON-O	AOR-O	AOS-O	AO2-O	APJ-O	APK-O
		RGB-O	RGL-O	ROI-O	ROJ-M	RON-O	RPC-O	RPF-O	RPM-U
PPIUNTCT	(304)	ABT-M	ABU-M	ABV-M	ABW-M	ABX-M	RBA-U	RBT-U	RBU-U
		RBV-U	RBW-U	RBX-U	RBY-U	RBZ-U	RON-U	8ON-U	
PPIVMV	(350)	ABF-U							
PPIVSI	(464)	RC6-M							
PPIVSMSG	(460)	AGA-M	AGI-M	AGK-M	AGN-M	APF-M	APK-M	RCT-U	RGV-U
		RPV-U	9GN-M						
PPIVSMSL	(45E)	AGA-M	AGI-M	AGK-M	AGN-M	APF-M	APK-M	RCT-M	RGV-U
		RPV-U	9GN-M						
PPIWKARE	(48)	ABF-M	AGD-O	AGF-M	AGL-O	AGM-O	AOI-M	APB-O	APE-O
		RCT-U	RCV-M	RGB-M	RGC-M	RGD-M	RGE-M	RGL-M	RGM-M
		RGV-M	ROI-M	ROJ-M	ROK-M	RON-M	ROR-M	ROS-M	ROT-M
		RPB-M	RPC-M	RPE-M	RPF-M	RPG-M	RPM-M	RPV-M	8GB-M
		8ON-M	8PA-M	8PM-M	9GC-O				
PPIWRT	(348)	ABC-U	APA-U	APB-U	APD-U	APE-U	RBA-U	RBB-U	RBC-U
		RBE-U	RBG-U	RBH-U	RBI-U	RBJ-U	RBK-U	RBL-U	RBM-U
		RBN-U	RBO-U	RBP-U	RBT-U	RBU-U	RBV-U	RBW-U	RBX-U
		RBY-U	RBZ-U	RC9-U	ROI-U	ROK-U	ROR-U	ROT-U	8BN-U
		8BO-U	8ON-U	9PA-U					
PPIXCAP	(268)	APD-U	ADQ-U	ADR-U	ADS-U	ADI-U	AON-U	RCJ-M	RC1-M
		RDD-U	RDE-U	RDP-M	RDQ-M	RDR-M	RDS-M	RDT-M	ROK-M
		RON-M	RPA-M	RPB-M	8PA-M				
PPIX11	(3B8)	APG-U	APL-U	EX1-O	RC6-M	RC7-O	RC9-O		
PPIX15	(3C8)	ADD-M	ADE-M	ADR-M	ADS-M	ADU-M	RC6-O	RC8-M	RDD-U
		RDE-U	RDR-U	RDS-U	RDU-U				
PPIX16	(410)	RCT-O	RC6-M	RDD-U	RDE-U	RDP-U	RDS-U		
PPIX17	(3D8)	RC6-O	RPC-U	RPM-U	8PM-U				
PPIX18	(3E8)	AGA-U	AGI-U	AGN-U	RC6-O	9GN-U			
PPIX19	(3F8)	AGA-U	AGI-U	AGN-U	RC6-O	9GN-U			
PPIX21	(3C0)	APH-U	APL-U	EX2-O	RC7-M				
PPIX25	(3D0)	RBJ-U	RBK-U	RBW-U	RBX-U	RC7-O			
PPIX27	(3E0)	RC7-O	RPF-U	RPM-U	8PM-U				
PPIX28	(3F0)	AGG-U	AGJ-U	AGN-U	RC7-O	9GN-U			
PPIX29	(400)	AGG-U	AGJ-U	AGN-U	RC7-O	9GN-U			
PPIX31	(3C0)	API-U	EX3-O	RC8-M					
PPIX35	(3D0)	ABM-M	ABO-M	RBM-U	RBO-U	RC8-O	8BO-U	9BO-M	
PPIX37	(3E0)	RC8-O	RPG-U						
PPIX38	(3F0)	AGK-U	APF-U	APK-U	RC8-O				
PPIX39	(400)	AGK-U	APF-U	APK-U	RC8-O				
PPIX61	(408)	AOB-U	AOD-U	AOF-U	AOH-U	AOQ-U	RC6-O	RC7-O	RC8-O

COMMON CROSS-REFERENCE TABLE (BLOCKSET)

The following table contains the names of COMMON fields together with their displacements.

Disp.	Name	Definition		Description
00EC	COMABEND	DS	C	ABDUMP DD ENCOUNTERED
0388	COMACE	DS	F	ABNORNAL CHANNEL END
05FD	COMALTAB	DS	CL256	
0250	COMAREAQ	DS	F	WORK AREA PREFERENCE QUEUE
01A8	COMARECS	DS	F	ADDED REC COUNT
0142	COMARKER	DS	C	CONTROL CARD ERROR POSITION FLAG
01E4	COMAVAIL	DS	F	ADDRESS OF AVAILABLE COMMON
0830	COMAVGLN	DS	F	AVERAGE RECORD LENGTH VALUE
0198	COMAXIO	DS	H	MAXIMUM SORTIO BUF TRANSFER COUNT
03A4	COMAXKPB	DS	H	MAXIMUM KEYS PER BLOCK
0118	COMAXLIM	DS	F	MAX FOR SIZE=MAX
090C	COMBACK	DS	F	BACKOUT RECORD'S PREDECESSOR
0254	COMBINFQ	DS	F	QUEUE OF AVAILABLE RECORD BINS
01C0	COMBINSN	DS	F	SELECTED BIN RECORD COUNT
0264	COMBINSQ	DS	F	SELECTED BIN QUEUE
0268	COMBINST	DS	F	SELECTED QUEUE TAIL
02AC	COMBINSZ	DS	F	SIZE OF A RECORD BIN
082A	COMBLKS	DS	C	N=BYPASS FLR BLOCKSET
020C	COMBLKSI	DS	F	WORKFILE BLOCK SIZE
041C	COMBLOKI	DS	F	INPUT BLOCK COUNTER
0418	COMBLOKO	DS	F	OUTPUT BLOCK COUNTER
0390	COMBOUND	DS	F	COUNT OF WITHIN BOUND RECORDS
03B4	COMBVCTR	DS	H	HIGH BV XFER COUNT
0374	COMBVD	DS	F	CURRENT BVD
027C	COMBVDAQ	DS	F	ALLOCATED BVD COUNT
0278	COMBVDFQ	DS	F	FREE BVD QUEUE
0284	COMBVDHI	DS	F	ADDRESS OF HIGH BOUNDARY VALUE DEF
0280	COMBVDLO	DS	F	ADDRESS OF LOW BOUNDARY VALUE DEF
03B6	COMBVLIM	DS	H	HIGH BV XFER LIMIT
0080	COMB37R	EQU	X'80'	B37 INFO MESSAGE FLAG MASK
0274	COMCADAQ	DS	F	AVAILABLE CYLINDER AREA QUEUE
0270	COMCADFQ	DS	F	FREE CAD QUEUE
02CC	COMCCHH	DS	F	LATEST SORTIO CCHH
08EC	COMCENT	DS	F	PER CENT RSA UTILIZATION
081D	COMCHALT	DS	CL1	A=TRANSLATE AQ ONLY C=TRANSLATE CH AND AQ
081E	COMCHECK	DS	CL1	Y=RECORD COUNTERS CHECKED N=RECORD COUNTERS NOT CHECKED
0484	COMCLOSE	DS	0F	CLOSE LIST AREA
082B	COMCNTL	DS	C	X'00'=SORTCNTL NOT PRESENT
00E8	COMCOBIN	DS	C	COBOL INPUT MODE OPTION
00E9	COMCOBUT	DS	C	COBOL OUTPUT MODE OPTION
0139	COMCPRNT	DS	C	COBOL SORT MESSAGE PRINT CLASS
0130	COMCSYSO	DS	CL8	COBOL SYSOUT=A DDNAME
03B0	COMCYLNO	DS	H	LOGICAL CYLINDER ASSIGNMENT CTR
08D4	COMDELTA	DS	F	RESIDUE DELTA
00F8	COMDIAGS	DS	F	DIAGNOSTIC LIST ADDRESS
01AC	COMDRECS	DS	F	DELETED REC COUNT
0358	COMDUBBL	DS	D	DOUBLEWORD ALIGNED WORK AREA
0248	COMDXKEY	DS	F	ADDRESS OF KEY AREA DXL
023C	COMDXLIO	DS	F	SORTIO CIRCULAR LIST ENTRY
0244	COMDXLRQ	DS	F	SORTWK DXL READ QUEUE
0238	COMDXLSQ	DS	F	SEQUENTIAL DXL LIST HEAD
02B4	COMDXLSZ	DS	F	SIZE OF DXL (DEPENDS ON V OR R)
0240	COMDXLWQ	DS	F	SORTWK DXL WRITE QUEUE
024C	COMDXSKY	DS	F	SECONDARY KEY AREA DXL ADDRESS
05F0	COMDYNAM	DS	CL8	DYNALLOC UNIT NAME
0814	COMDYNER	DS	F	RETURN CODE FROM DYNALLOC
05F8	COMDYNUM	DS	CL8	DYNALLOC COUNT
092C	COMECTRS	DS	10F	EVENT COUNT SAVE AREA
0128	COMEMBRI	DS	CL8	SYSOUT FILE NAME

Disp.	Name	Definition		Description
0130	COMEMBRO	DS	CL8	COBOL SYSOUT=A DDNAME
00F2	COMEOF	DS	C	END OF FILE ON SORTIN INDICATOR
00F1	COMEOV	DS	C	END OF VOLUME ON SORTIN INDICATOR
02A4	COMEQUAL	DS	F	EQUAL COUNTER
0484	COMERGEI	DS	28F	
04F4	COMERGES	DS	48F	
0920	COMESAVE	DS	3F	EVENT COUNT SAVE AREA
02A8	COMEXITS	DS	F	EXIT LIST HEAD
0960	COMEXTRA	DS	2F	UNUSED WORDS
00E4	COMFAULT	DS	0F	
0968	COMFEND	DS	0D	END OF FOUNDATION
01D4	COMFILEG	DS	F	GET FILE AREA PTR
01D8	COMFILEP	DS	F	PUT FILE AREA PTR
019C	COMFILEZ	DS	F	FILE SIZE
0010	COMFLAG	EQU	16	TRACE INPUT BLOCKS
08D0	COMFLOOR	DS	F	PAGE SIZE COMPLEMENT
00E4	COMFOUND	DS	0F	
01BC	COMFREEB	DS	F	FREE BIN COUNT
08E8	COMFSTOR	DS	F	FIXED RESTORE ROUTINE ADDRESS
0420	COMHASH	DS	F	SUM OF OUTPUT COUNT IDS
0145	COMIBEND	DS	C	ABEND IF ERROR DURING INITIAL PHASE
0192	COMIBLKL	DS	H	INPUT FILE AVERAGE BLOCK LENGTH
018A	COMIBLKZ	DS	H	INPUT BLOCKSIZE
036C	COMIDKEY	DS	F	MIDDLE KEY
0484	COMIDSNM	EQU	COMCLOSE	INPUT DATA SET NAME
0146	COMIDUMP	DS	C	DUMP OPTION FOR IBEND
0188	COMIFORM	DS	C	INPUT REDFM
0380	COMIIOB	DS	F	INPUT IOB
01F4	COMINCOR	DS	F	MINIMUM SPACE NEEDED TO SORT
0154	COMINDCB	DS	F	SORTIN DCB ADDRESS
0918	COMINFIL	DS	F	SORTIN BYTE COUNT
01F0	COMINGET	DS	F	MINIMUM GETMAIN AMOUNT
0196	COMINIO	DS	H	MINIMUM SORTIO BUF TRANSFER COUNT
03A2	COMINKPB	DS	H	MINIMUM KEYS PER BLOCK
00EF	COMINOUT	DS	C	COMMON IN/OUT DATA SET
08D8	COMINSEG	DS	F	MIN SIZE FOR 1ST SEGMENT
00FC	COMINUS4	DS	F	MINUS 4
0100	COMINUS8	DS	F	MINUS 8
0260	COMIOBXQ	DS	F	QUEUE OF EXCP IOBS
0230	COMIOLOC	DS	F	SORTIO DATA ADDRESS IN IOX
02BC	COMIORPB	DS	F	SORTIO REC PER BLOCK
00F0	COMIOTYP	DS	C	V=VIRTUAL, R=REAL
0224	COMIOXAD	DS	F	ADDRESS OF IOX
0228	COMIOXSZ	DS	F	LENGTH OF IOX
01A0	COMIRECS	DS	F	INPUT REC COUNT
018E	COMIRECZ	DS	H	INPUT LRECL
08C0	COMIRES	DS	F	START OF RESIDUAL INPUT DATA
01CC	COMIRSAZ	DS	F	INPUT RSA SIZE
0370	COMISCEL	DS	4C	SWITCH VALUES
0001	COMISREL	EQU	1	0 => RELEASE SORTWORK SPACE
0143	COMJOB	DS	C	TYPE/LOG JOB NAME
0220	COMKBLKZ	DS	F	SIZE OF KAD BLOCK ENTRY
021C	COMKDATA	DS	F	SIZE OF KAD PREFIX
03AA	COMKDELT	DS	H	PRIMARY KEY MULTIPLE LOAD DELTA
026C	COMKEYFQ	DS	F	FREE KEY HOLDER QUEUE
0398	COMKEYPC	DS	H	PRIMARY KEY COUNT
039C	COMKEYPN	DS	H	PRIMARY KEY MINIMUM
0288	COMKEYPQ	DS	F	PRIMARY KEY QUEUE
028C	COMKEYPT	DS	F	PRIMARY KEY QUEUE TAIL
039E	COMKEYPX	DS	H	PRIMARY KEY MAXIMUM
039A	COMKEYSC	DS	H	SECONDARY KEY COUNT
0290	COMKEYSQ	DS	F	SECONDARY KEY QUEUE
03A0	COMKEYSX	DS	H	SECONDARY KEY MAXIMUM
0294	COMKIRRF	DS	F	KAD IRRR HOLDER FREE QUEUE
0298	COMKIRRQ	DS	F	KAD IRRR HOLDER QUEUE

Disp.	Name	Definition		Description
0174	COMKLOAD	DS	F	KEY LOAD SUBROUTINE ADDRESS
01D0	COMKOUNT	DS	F	PRIMARY KAD COUNT
0214	COMKPB	DS	F	KEYS PER WORKFILE BLOCK
0218	COMKPFIX	DS	F	SIZE OF BIN KAD PREFIX
03A6	COMKRECZ	DS	H	KEY RECORD LENGTH
0954	COMKSAVE	DS	F	KEY SAVE AREA ADDRESS
0178	COMKUNLD	DS	F	KEY UNLOAD SUBROUTINE ADDRESS
080C	COMLADAD	DS	F	BLOCKED LAD ADDR/ADDR (E35)
0258	COMLADFQ	DS	F	LAD AVAILABLE LIST
037C	COMLADLM	DS	F	LAD AREA LIMIT
025C	COMLADNQ	DS	F	NEXT STRING LAD QUEUE
0378	COMLADS	DS	F	LAD AREA ADDRESS
013C	COMLCTR	DS	C	INITIAL LINE COUNTER VALUE
0180	COMLDKAD	DS	F	ADDRESS OF KAD LOAD ROUTINE
08F8	COMLEND	DS	F	UNBALANCED LIST END
0394	COMLIMIT	DS	F	BOUND LIMIT
013D	COMLINES	DS	C	LINES PER PAGE
08F4	COMLIST	DS	F	UNBALANCED LIST HEAD
01FC	COMLISTA	DS	F	LIST HEADER FOR MERGE SORT
0200	COMLISTB	DS	F	LIST HEADER FOR MERGE SORT
016C	COMLOAD	DS	F	LOAD SUBROUTINE ADDRESS
042C	COMLOCAL	DS	22F	
013B	COMLOG	DS	C	LOG MESSAGE CLASS
0900	COMLOKEY	DS	F	POINTER TO LOW LIMIT KEY AREA
022C	COMLOWIO	DS	F	LOW I/O ADDRESS
0149	COMLPA	DS	C	IPUT AND OPUT ARE IN LPA
095C	COMLSTAB	DS	F	LENGTH STATISTICS TABLE
03AC	COMLSTAR	DS	H	1ST TWO BYTES OF JFCLSTAR
0958	COMLSTAT	DS	F	LENGTH STATISTICS CTR PTR
011C	COMMAIN	DS	F	MAX FOR SIZE= NOT SPECIFIED
01BC	COMMAIN0	DS	0F	SAVED INIT MAIN STORAGE VALUE
01C0	COMMAIN1	DS	F	SELECTED BIN RECORD COUNT
01C8	COMMAIN2	DS	0F	USED MAIN STORAGE VALUE
01DC	COMMAREA	DS	F	ADDRESS OF COMMON WORK AREA
01E0	COMMEND	DS	F	ADDRESS OF END OF COMMON WORK AREA
00EB	COMMNOGO	DS	C	ERROR ENCOUNTERED INDICATOR
0000	COMMON	DS	3F	START OF OS SAVE AREA
08BC	COMMSGPR	DS	H	MESSAGE PARAMETER SAVE AREA
0828	COMMSGSW	DS	X	INFORMATION MSG FLAG BYTE
0020	COMMSNAP	EQU	32	SNAP WRITE BACK BLOCKS
00EE	COMMTEST	DS	C	TEST STATUS INDICATOR
0008	COMMTPH0	EQU	8	ISSUE PHASE 0 MESSAGES
0040	COMMTSEQ	EQU	64	SEQUENCE CHECK SORTOUT
0020	COMMTVER	EQU	32	VERIFY BIN COUNT
0080	COMMT190	EQU	128	ISSUE 190 MESSAGES
0968	COMMVARY	EQU	*	BEGINNING OF VARIABLE AREA
0820	COMMXBLK	DS	F	MAX SORTIN BLKSIZE FOR SMF
0104	COMMXCTL	DS	CL8	NAME TO XCTL TO
0428	COMNMAX	DS	F	ESTIMATED NMAX VALUE
0147	COMOBEND	DS	C	ABEND IF ERROR DURING OTHER PHASE
018C	COMOBLKZ	DS	H	OUTPUT BLOCKSIZE
0148	COMODUMP	DS	C	DUMP OPTION FOR OBEND
0189	COMOFORM	DS	C	OUTPUT RECFM
0384	COMOIOB	DS	F	OUTPUT IOB
0808	COMOJFCB	DS	F	ADDRESS OF SORTOUT JFCB
0364	COMOLEND	DS	F	END OF OPEN LIST
0360	COMOLIST	DS	F	OPEN LIST ADDRESS
0834	COMOPLOC	DS	20F	OPTION STATEMENT WORK AREA
0104	COMOPTNS	DS	0F	
0048	COMOPTZ	EQU	*-COMOPTNS	SIZE OF OPTIONS
010C	COMOPWTO	DS	CL12	WTO HEADER AND ROUTINE
01A4	COMORECS	DS	F	OUTPUT REC COUNT
0190	COMORECZ	DS	H	OUTPUT LRECL
08C8	COMORES	DS	F	START OF RESIDUAL OUTPUT DATA

Disp.	Name	Definition		Description
029C	COMOVALT	DS	F	ALTERNATE REC MOVE
0400	COMOVKAD	DS	CL18	MVC 0(KADLIST,R1),0(R15)
03BE	COMOVKEY	DS	CL6	MVC 0(,R1),0(R15)
017C	COMOVREC	DS	F	ADDRESS OF RECORD MOVE ROUTINE
03C4	COMPAB	DS	CL6	CLC 8(,R4),0(R3)
03F4	COMPALTA	DS	CL6	CLC 4(,R2),0(R4)
03FA	COMPALTB	DS	CL6	CLC 4(,R3),0(R4)
03E8	COMPARE	DS	F	CLC 0(,R14),0(R15)
02B8	COMPAREZ	DS	F	LENGTH OF COMPARE
0908	COMPARZ6	DS	F	OFFSET IN BIN TO SEG BYTE
03CA	COMPCOPA	DS	CL6	CLC 4(,R4),0(R14)
03D0	COMPCOPB	DS	CL6	CLC 4(,R5),0(R14)
03D6	COMPCOPC	DS	CL6	CLC 0(,R4),0(R14)
08FC	COMPEERS	DS	F	PEERAGE ACCUMULATOR
013F	COMPFIX	DS	CL3	PREFIX NAME
03E2	COMPICK	DS	CL6	CLC 0(,R2),BVDKEY-BVDSECT(R4)
07F0	COMPLADS	DS	6F	
07EC	COMPLAST	DS	F	LAST LIST ENTRY
0774	COMPLIST	DS	30F	
07F0	COMPLONG	DS	6F	
00F4	COMPMODE	DS	C	'L'=LOW, 'H'=HIGH
03B8	COMPMOVE	DS	0H	
03DC	COMPPOUND	DS	CL6	CLC 0(,R2),0(R8)
014C	COMPRDCB	DS	F	ADDRESS OF SYSOUT DCB
0164	COMPRDEC	DS	F	ADDRESS OF SYSOUT DEC
0138	COMPRINT	DS	C	PRINT MESSAGE CLASS
03B2	COMPRIZE	DS	H	PRIOR BLOCK SIZE
03EE	COMPSLAB	DS	CL8	CLC 4(,R4),4(R5)
03CA	COMPSLAK	DS	CL6	CLC 4(,R4),0(R14)
03D0	COMPSLBK	DS	CL6	CLC 4(,R5),0(R14)
0150	COMRDDCB	DS	F	ADDRESS OF SYSIN DCB
0168	COMRDDEC	DS	F	SYSIN DECB ADDRESS
00EA	COMRECFM	DS	C	SORT RECORD FORM
02D0	COMRECNO	DS	2C	LATEST R
0210	COMRPB	DS	F	RECORDS PER WORKFILE BLOCK
01C8	COMRSASZ	DS	F	RSA SIZE
08B4	COMRSAV1	DS	F	REGISTER SAVE AREA
08B8	COMRSAV2	DS	F	REGISTER AREA
08E0	COMRSAWK	DS	F	RSA TO WORK BUF ROUTINE
08E4	COMRSTOR	DS	F	RESTORE LIST POINTER
0810	COMRTAIL	DS	F	LAST FOUR BYTES OF RECORD
01C4	COMRUNS	DS	F	RUNS DISTRIBUTED BY INPUT PHASE
01B0	COMRUNSZ	DS	F	CURRENT RUN LENGTH
01E8	COMSAD	DS	F	ADDRESS OF FIRST SOTRAGE AREA DESC
02C4	COMSAVEI	DS	F	TIOT ENTRY FOR SORTIN
02C8	COMSAVEO	DS	F	TIOT ENTRY FOR SORTOUT
000C	COMSAVE0	DS	15F	LEVEL ZERO SAVE AREA
0048	COMSAVE1	DS	13F	LEVEL ONE SAVE AREA
007C	COMSAVE2	DS	13F	LEVEL TWO SAVE AREA
00B0	COMSAVE3	DS	13F	LEVEL THREE SAVE AREA
0120	COMSERVE	DS	F	AMOUNT OF RESERVED MEMORY
013E	COMSGLST	DS	C	LIST CONTROL CARDS OPTION
0910	COMSGTOT	DS	F	TOTAL BYTES SEGMENTED
01F8	COMSHORT	DS	F	MAIN STORAGE DEFICIT
038C	COMSIRRR	DS	F	SECONDARY KEYFILE IRRR
00F3	COMSKEYS	DS	C	SECONDARY KEEP PRODUCED
00F5	COMSKIPM	DS	C	'Y'=SKIP INTERMEDIATE MERGE
081F	COMSMF	DS	C	'S'=SHORT SMF RECORD 'F'=FULL SMF RECORD (NOT VALID FOR FLR) 'N'=NO SMF RECORD
0124	COMSPFIX	DS	CL4	'SORT' DD PREFIX OR ALIAS
0368	COMSQRT	DS	F	SQUARE ROOT OF RPB
0144	COMSTEP	DS	C	TYPE/LOG STEP NAME

Disp.	Name	Definition		Description
01EC	COMSPACE	DS	F	BYTES AVAILABLE
0184	COMSTORE	DS	F	RESTORE CODE ADDRESS
014A	COMSVC	DS	H	SVC FOR SORT
0818	COMSYSRV	DS	F	SYSTEM RESERVED
00ED	COMSYSTM	DS	C	SYSTEM TYPE FROM CVT
0128	COMSYSUT	DS	CL8	SYSOUT FILE NAME
04F0	COMTABLE	EQU	COMERGES - 4	TRANSLATE/SIZE TABLE
02C0	COMTIOT	DS	F	ADDRESS OF TIOT
0904	COMTOTAL	DS	F	TOTAL RSA BYTES NOT IN OUTPUT RUN
0834	COMTRACE	DS	1F	TRACE X'01'-X'0F' E15/E35 IN/OUT AREA RECOR LEVEL
082C	COMTRACK	DS	F	#TRACKS ALLOCATED FOR SORTWK
02A0	COMTRACT	DS	F	EXTRACT ADDRESS
081C	COMTRANS	DS	X	TRANSFER INDICATION FOR ICEMESS
02D2	COMTRBAL	DS	H	CURRENT TRACK BALANCE
0204	COMTREE	DS	F	40F
0644	COMTREEA	DS	40F	
06E4	COMTREED	DS	16F	
05B4	COMTREEI	DS	36F	
0724	COMTREES	DS	20F	
02D4	COMTRSAV	DS	H	PREVIOUS TRACK BALANCE
0208	COMTSIZE	DS	F	OFFSET TO LAST NODE
0824	COMTTIME	DS	F	CPU TIME WORK AREA FOR SMF
013A	COMTYPE	DS	C	TYPE MESSAGE CLASS
08F0	COMUNBAL	DS	F	ADDRESS OF UNBALANCED MERGE
0170	COMUNLD	DS	F	UNLOAD SUBROUTINE ADDRESS
0158	COMUTDCB	DS	F	SORTOUT DCB ADDRESS
091C	COMUTFIL	DS	F	SORTOUT BYTE COUNT
01B4	COMVALCT	DS	F	BVAL STANDARD COUNT
01B8	COMVALPR	DS	F	BVAL PRIOR COUNT
0829	COMVBLKS	DS	C	N=BYPASS VLR BLOCKSET
0003	COMVERSE	EQU	3	CURRENT COMMON VERSION NUMBER
00E4	COMVERZN	DS	H	CONTAINS COMVERSE
05FC	COMVIO	DC	C	VIO/NOVIO SWITCH
00E6	COMVOKED	DS	C	USER INVOKED STATUS
0194	COMVOLCT	DS	C	SORTIN VOLUME COUNT
03AE	COMWDXLS	DS	H	WORKFILE DXLS
02E4	COMWJOB	EQU	COMWMTAG + 2	JOB NAME
015C	COMWKDCB	DS	F	SORTWK OPEN LIST ADDRESS
0914	COMWKFIL	DS	F	BYTES IN WORK FILE
0234	COMWKLOC	DS	F	SORTWK DATA ADDRESS IN IOX
08DC	COMWKRSA	DS	F	WORK BUF TO RSA ROUTINE
08CC	COMWKSIZ	DS	F	SIZE OF WORK AREA
02F6	COMWMESS	EQU	COMWSTEP + 9	MESSAGE
02DF	COMWMNUM	EQU	COMWPFIX + 3	MESSAGE NUMBER
02E2	COMWMTAG	EQU	COMWMNUM + 3	MESSAGE TAG
02DC	COMWPFIX	EQU	COMWTEXT	MESSAGE PREFIX
02B0	COMWRECZ	DS	F	SIZE OF A WORKFILE RECORD
08C4	COMWRES	DS	F	START OF RESIDUAL WORK AREA
0354	COMWROUT	DS	F	ROUTINE FOR WTO
02ED	COMWSTEP	EQU	COMWJOB + 9	STEP NAME
02DC	COMWTEXT	CL120		TEXT OF WTO
02D8	COMWTO	DS	F	HEADER FOR L FORM OF WTO
03B8	COMXCRUN	DS	CL6	XC 0(,R1),0(R1)
0004	COMXE01	EQU	4	E01 IS PRESENT
0010	COMXE03	EQU	16	E03 IS PRESENT
0001	COMXE15	EQU	1	E15 IS PRESENT
0002	COMXE35	EQU	2	E35 IS PRESENT
0371	COMXSTAT	EQU	COMISCEL + 1	EXIT STATUS
02D6	COMXTENT	DS	H	CURRENT EXTENT
0160	COM01DCB	DS	F	SORTIN01 OPEN LIST ADDRESS
0424	COM2NDRY	DS	F	SECONDARY ALLOCATION WORD
0724	COM43LST	DS	0F	LIST AREA FOR CON043A MSGS

COMMA CROSS-REFERENCE TABLE (PEERAGE/VALE)

The following table contains the names of the COMMA fields together with their displacements.

DISP	LABEL	ATTR	COMMENT
5E5	CABRETRN	BIT(1)	TO PREV SEC. ENTR OF ABENDR
188	CADD	PTR(31)	ADDR. OF TREEADD ROUTINE
188	CADDISP	FIXED(31)	TREEGEN WORK CONSTANT
18C	CADDS	PTR(31)	ADDR. OF TREEADDS ROUTINE
3F4	CADDTAB	FIXED(31)	TREEGEN WORK AREA
495	CALLOCND	BIT(1)	ALL WORKFILES ALLOCATED
80C	CAVGRCDL	FIXED(31)	AVGERAGE RECORD LENGTH
154	CBALADDR	PTR(31)	ADDR. OF ROUTINE BALLOCAT
190	CBASEMIN	FIXED(15)	BASE-BIN MIN ALLOC
3A4	CBASESIZ	FIXED(31)	BASE-BIN SIZE @PTM411M
194	CBINBLK	PTR(31)	ADDR TO BINBLK Q
198	CBINSIZE	PTR(31)	RSA BIN SIZE FOR MERGING
7EC	CBLDLNAM	CHAR(8)	MEMBER NAME FOR SORTIN
3E4	CBLKIN	FIXED(15)	INPUT BLOCKSIZE
19C	CBLKITTR	PTR(31)	NEXT UNUSED ITTR
1A4	CBLKLSIZ	FIXED(31)	SIZE OF FREE BLOCK LIST
6A0	CBLKPTRK	FIXED(31)	BLKS/TRK LOACALLY
346	CBLKSET	CHAR(1)	Y=BLOCKSET MAY BE USED
1F4	CBRANCHP	PTR(31)	ADDR. OF TREEGEN BR-TABLE
1A0	CBUFFER	PTR(31)	ADDR. OF CURRENT WORK BUF
1A8	CBUFPTR	PTR(31)	IN - PTR IN BUFF TO VLR
7AC	CBUGADDR	FIXED(31)	ICEBUG AREA ADDRESSES
5E5	CB37MSG	BIT(1)	B37 RECOVERY MAG FLAG @L5A
347	CCHALT	CHAR(1)	C=TRANSLATE CH AND AQ
34A	CCHECK	CHAR(1)	Y=CHECK COUNTERS
4A0	CCHECKPT	CHAR(1)	YES = TAKE CHECKPOINTS
1AC	CCHIDX1	FIXED(31)	INDEX AREA 1 HEAD
1B0	CCHIDX2	FIXED(31)	INDEX AREA 2 HEAD
1B4	CCHIDX3	FIXED(31)	INDEX POOL HEAD
1B8	CCHIDX4	FIXED(31)	POOL AREA 1 HEAD
1BC	CCHIDX5	FIXED(31)	POOL AREA 2 HEAD
495	CCONCASW	BIT(1)	CONCATENATION OF DATASETS
4A3	CCONMODE	CHAR(1)	Y=PEER N=NONPEER 0=5734
1C4	CCONNEG4	FIXED(31)	CONSTANT = -4
330	CCONSMOD	CHAR(1)	Y = MESSAGES TO CONSOLE
1C0	CCON256	FIXED(31)	CONSTANT = 256
1C8	CCORESAV	FIXED(31)	ORIGINAL SIZE PARM
333	CCRITMOD	CHAR(1)	Y = PRINT CRITICAL MSG
1D4	CCURRBLK	PTR(31)	ADDR TO CURR BINBLK
150	CDALADDR	PTR(31)	ADDR. OF ROUTINE DALLOCAT
1E4	CDATLEN	FIXED(31)	DATA BLOCK SIZE
1DC	CDATLOC	BDY(WORD)	ICEMAN BASEREGS
1D8	CDCBLIST	PTR(31)	ADDR. OF OPEN/CLOSE LIST
494	CDDFIRST	BIT(1)	PROC.FIRST DDNAME CHARACTER
21E	CDEBUG	CHAR(1)	DEBUG SWITCH BYTE
1E8	CDECAREA	PTR(31)	ADDR. OF SORTIN/OUT DECB
784	CDECAREB	PTR(31)	USED IN ICELINK
588	CDECDED	PTR(31)	ICEDEC - ICEDED COMM
57C	CDECL1	FIXED(31)	REC LEN1 IN DEC
580	CDECL2	FIXED(31)	REC LEN2 IN DEC
584	CDECL3	FIXED(31)	REC LEN3 IN DEC
1EC	CDECSIZE	FIXED(31)	SIZE OF SORTIN/OUT DECB
1F0	CDELCNT	FIXED(31)	BYTES DELETE IN RSA
1F4	CDELETE	PTR(31)	ADDR. OF TREELETE ROUTINE
588	CDEVWK	PTR(31)	PTR TO DEV WORK AREA
1F8	CDEXCTR	FIXED(31)	HI/LO INDEX COUNTER
1FC	CDEXITTR	PTR(31)	INDEX BLOCK ITTR

DISP	LABEL	ATTR	COMMENT
200	CDEXPBUF	FIXED(31)	INDEX ENTRIES PER BUFFER
204	CDEXPTRK	FIXED(31)	INDEX ENTRIES PER TRACK
49E	CDIAGMOD	CHAR(1)	YES = PRINT DIAGNOSTICS
218	CDIMSHI	FIXED(16)	HI CONTROL FIELD FOR IMS
7E2	CDISPMOD	CHAR(1)	DISP IS MOD FOR SORTOUT
496	CDKALLOC	BIT(1)	SORTDK ALLOCATED
6EC	CDOT	CHAR(1)	DOT
208	CDYNAMIC	PTR(31)	ADDR OF UNUSED DYNAMIC AREA
496	CDYNGEN	BIT(1)	DYNALLOC GENERIC NAMES USED
21B	CDYNLEN	PTR(8)	DYNALLOC FIELD LENGTH
214	CDYNNUMB	FIXED(31)	NUMB OF DYN ALLOC DEVICES
495	CDYNSW	BIT(1)	ONE DYNALLOC FAILED
20C	CDYNWK	CHAR(8)	DYN ALLOC DEVICE TYPE
20C	CDYNWK1	PTR(8)	1ST BYTE OF DYN DEV TYPE
21C	CEBUFFRS	FIXED(16)	NUMBER OF E-PHASE BUFFERS
228	CEMERGE	FIXED(31)	MERGE ORDER - ELIMINATION
21F	CENDMODE	CHAR(1)	NO = UNEXPECTED SYSIN EOF
178	CEQDISPL	PTR(31)	WORK AREA FOR EXTRACT
794	CEQE25	PTR(31)	PICK UP EQUALS FIELD
794	CEQE25OP	CHAR(2)	OP CODE ICM
334	CEQUALS	CHAR(1)	Y = PRESERV ORDER OF EQUALS
337	CERRINV	CHAR(1)	Y = CRITICAL ERROR ABEND
494	CERRSW1	BIT(1)	ERROR IN REQUIRED PARAMETER
494	CERRSW2	BIT(1)	ERROR IN OPTIONAL PARAMETER
336	CERRU016	CHAR(1)	Y = CRITICAL ERROR ABEND
4A8	CESTIMSZ	CHAR(1)	YES = ESTIM. SIZE|FILSZ
4A7	CEXACTSZ	CHAR(1)	YES = EXACT FILSZ GIVEN
7AA	CEXCPIN	CHAR(1)	EXCP CAN BE USED FOR SORTIN
7AB	CEXCPOUT	CHAR(1)	EXCP -//- SORTOUT
790	CEXITDEL	PTR(31)	EXIT DELETE ROUTINE ADDR.
78C	CEXITLOA	PTR(31)	EXIT LOAD ROUTINE ADDRESS
22C	CEXITPH1	FIXED(31)	LNG OF PH1 RTNS
230	CEXITPH2	FIXED(31)	LNG OF PH2 RTNS
234	CEXITPH3	FIXED(31)	LNG OF PH3 RTNS
57C	CEXITTAB	CHAR(16)	EXIT RETURN BRANCHTAB
23C	CEXIT11	BDY(WORD)	E11 EXIT-
248	CEXIT15	BDY(WORD)	E15/E32 EXIT -
254	CEXIT16	BDY(WORD)	E16 EXIT
260	CEXIT17	BDY(WORD)	E17 EXIT
26C	CEXIT18	BDY(WORD)	E18 EXIT
284	CEXIT19	BDY(WORD)	E19 EXIT
29C	CEXIT21	BDY(WORD)	E21 EXIT
768	CEXIT25	BDY(WORD)	EXIT25
2A8	CEXIT27	BDY(WORD)	E27 EXIT
2B4	CEXIT31	BDY(WORD)	E31 EXIT
2C0	CEXIT35	BDY(WORD)	E35 EXIT -
2CC	CEXIT37	BDY(WORD)	CE37 EXIT
2D8	CEXIT38	BDY(WORD)	E38 EXIT
2E4	CEXIT39	BDY(WORD)	E39 EXIT
290	CEXIT61	BDY(WORD)	E61 EXIT
341	CEXMODE	CHAR(1)	R=EXCPVR V=EXCP X=MVT
224	CEXPAND	FIXED(31)	INT LRECL EXPAND POS OR NEG
220	CEXTRACT	PTR(31)	ADDR. OF EXTRACT ROUTINE
244	CE11CALL	PTR(31)	CALL ADDRES
244	CE11LNG	FIXED(31)	EXIT LENGTH
23C	CE11NAME	CHAR(8)	NAME
248	CE15ADDR	PTR(31)	PASSED ADDRESS
250	CE15CALL	PTR(31)	CALL ADDRESS
6D4	CE15DELT	BIT(1)	1=E15 DELETED
250	CE15LNG	FIXED(31)	LENGTH
4A4	CE15MOD	CHAR(1)	YES = E15 ADDRESS GIVEN
248	CE15NAME	CHAR(8)	NAME
58C	CE15PARM	CHAR(4)	E15 PARAMEYER LIST
25C	CE16CALL	PTR(31)	CALL ADDR
25C	CE16LNG	FIXED(31)	E16 LENGTH

DISP	LABEL	ATTR	COMMENT
254	CE16NAME	CHAR(8)	NAME
268	CE17CALL	PTR(31)	CALL ADDRESS
268	CE17LNG	FIXED(31)	E17 LENGTH
260	CE17NAME	CHAR(8)	NAME
274	CE18CALL	PTR(31)	CALL ADDRESS
238	CE18EOD	PTR(31)	USER EOD ADDR
27C	CE18EXL	PTR(31)	ADDR TO VSAM EXIT LIST
274	CE18LNG	FIXED(31)	E18 LENGTH
26C	CE18NAME	CHAR(8)	NAME
280	CE18PSW	PTR(31)	ADDR TO VSAM PASSW. LIST
278	CE18XLST	PTR(31)	ADDR TO USER XLIST(5) RTN
28C	CE19CALL	PTR(31)	CALL ADDRESS
28C	CE19LNG	FIXED(31)	E19 LENGTH
284	CE19NAME	CHAR(8)	NAME
2A4	CE21CALL	PTR(31)	CALL ADDR
2A4	CE21LNG	FIXED(31)	E21 LENGTH
29C	CE21NAME	CHAR(8)	NAME
770	CE25CALL	PTR(31)	AND ITS CALL ADDRESS
770	CE25LNG	FIXED(31)	EXIT E25 LENGTH
768	CE25NAME	CHAR(8)	EXIT E25 NAME
2B0	CE27CALL	PTR(31)	CALL ADDR
2B0	CE27LNG	FIXED(31)	E27 LENGTH
2A8	CE27NAME	CHAR(8)	NAME
2BC	CE31CALL	PTR(31)	CALL ADDRESS
2BC	CE31LNG	FIXED(31)	E31 LENGTH
2B4	CE31NAME	CHAR(8)	NAME
24C	CE32MO	FIXED(31)	E32 MERGE ORDER
24D	CE32MO24	PTR(24)	
2C0	CE35ADDR	PTR(31)	PASSED ADDRESS
5F4	CE35AREA	PTR(31)	
2C8	CE35CALL	PTR(31)	CALL ADDRESS
2C8	CE35LNG	FIXED(31)	LENGTH
4A5	CE35MOD	CHAR(1)	YES = E35 ADDRESS GIVEN
2C0	CE35NAME	CHAR(8)	NAME
590	CE35PARM	CHAR(12)	E35 PARAMETER LIST
2D4	CE37CALL	PTR(31)	CALL ADDRESS
2D4	CE37LNG	FIXED(31)	E37 LENGTH
2CC	CE37NAME	CHAR(8)	NAME
2E0	CE38CALL	PTR(31)	CALL ADDRESS
2E0	CE38LNG	FIXED(31)	E38 LENGTH
2D8	CE38NAME	CHAR(8)	NAME
2EC	CE39CALL	PTR(31)	CALL ADDRESS
2F0	CE39EXL	PTR(31)	ADDR TO VSAM EXIT LIST
2EC	CE39LNG	FIXED(31)	E39 LENGTH
2E4	CE39NAME	CHAR(8)	NAME
2F4	CE39PSW	PTR(31)	ADDR TO VSAM PASSW. LIST
2F8	CE39XLST	PTR(31)	ADDR TO USER XLIST(5) RTN
298	CE61CALL	PTR(31)	CALL ADDR
298	CE61LNG	FIXED(31)	E61 LENGTH
290	CE61NAME	CHAR(8)	NAME
	CE61PADD	FIXED(31)	REC ADDRESS
2FC	CE61PARM		E61 PARM LIST
2FF	CE61PCF	CHAR(1)	CONTR FIELD NUMB
	CE61PLEN	FIXED(31)	CONTR FIELD LEN
496	CFAKE35	BIT(1)	L2 GET L3 SWITCH
308	CFIELDS	FIXED(31)	NUMBER OF CONTROL FIELDS
804	CFILBYTE	FIXED(31)	FILE SIZE ACCUM (BYTES)
310	CFILIMIT	PTR(31)	ADDR. OF END OF BUFFER
30C	CFILSIZE	FIXED(31)	USER SPECIFIED FILE SIZE
30D	CFILSZ24	PTR(24)	THREE BYTES
495	CFIRSTSW	BIT(1)	OPEN OF FIRST DATA SET
	CFIRSTSB	BIT(1)	1=FIRST SUB-BIN
495	CFIXPAGE	BIT(1)	PROTECTED AREA FIXED
5D7	CFIXREC	BIT(1)	ON = FIXED LENGTH RECORDS
314	CFREECTR	FIXED(31)	AVAILABLE RSA BIN COUNT

DISP	LABEL	ATTR	COMMENT
180	CFREHEAD	PTR(31)	ADDR. OF FIRST FREE AREA
180	CFRELIST	BDY(DWORD)	START OF FREE SPACE CHAIN
144	CFRESTOR	PTR(31)	ADDR. OF ROUTINE FREESTOR
184	CFREZERO	FIXED(31)	SIZE OF ZEROTH FREE AREA
6D4	CFRSTBIN	BIT(1)	1=FIRST BIN ALLOC
148	CGALADDR	PTR(31)	ADDR. OF ROUTINE GALLOCAT
320	CGENDOPT	BDY(WORD)	SORT GENERATED DEFAULTS:
32C	CGENDOP1	BDY(WORD)	SORT GEN DEFAULTS CONT:
140	CGETMAIN	PTR(31)	ADDR. OF ROUTINE GETMAINS
354	CGETMTMP	PTR(31)	ADDR. OF TEMP. GETM. AREAS
35C	CGETPRM	PTR(31)	PERM AREAS CHAIN
358	CGETREC	PTR(31)	ADDR. OF INPUT GET ROUTINE
354	CGETTMP	PTR(31)	ALTERNATE NAME
318	CGMINDEX	FIXED(31)	INDEX IN GETMAIN TABLE
368	CHBUF1	FIXED(31)	INDEX HDR BFR ADDR
360	CHECKDCB	FIXED(31)	CHECKPOINT DCB POINTER
36C	CHKBLK	FIXED(31)	TOTAL BLKS EST FOR PRIM EXT
364	CHKTTSTR	PTR(31)	CHKPT ITT SAVE AREA
378	CIBSPACE	FIXED(31)	INPUT BFR SPACE
374	CIBUFFER	PTR(31)	ADDR. OF INPUT INDEX BUFFER
384	CIDEXSIZ	FIXED(31)	INDEX ENTRY SIZE
390	CIDXLEN	FIXED(31)	INDEX BLOCK SIZE
352	CIDXPLB	FIXED(15)	BLOCKS/LOGIC INDEX
495	CIMS	BIT(1)	SORT INVOKED BY IMS
496	CINCHAIN	BIT(1)	NO CHAIN SCHED FOR SORTIN
7F8	CINCONCA	CHAR(1)	CONCATENATED SORTIN FLAG
6D4	CINCORE	BIT(1)	1=INCORE SORT
37C	CINDEX	PTR(31)	ADDR. OF INPUT INDEX ENTRY
380	CINDEXHI	PTR(31)	ADDR. OF END OF INDEX BUF
780	CINDEXH1	PTR(31)	
77C	CINDEX1	PTR(31)	BUFFER POINTERS
394	CINLOC	FIXED(31)	ADDR TO INPUT REC
370	CINSIZE	FIXED(31)	SORTIN LOGICAL RCD LENGTH
18C	CINSTRLT	FIXED(31)	TREEGEN WORK AREA
49F	CINVOKED	CHAR(1)	YES = DYNAMIC INVOCATION
1CC	CIOHOLD	FIXED(31)	HOLD SW/Q END
1CC	CIOINIT	FIXED(31)	ADDR OF INIT ROUT
15C	CIORADDR	PTR(31)	ADDR. OF ROUTINE INOUTRTN
1D0	CIOSAV13	FIXED(31)	I/O TASK SAVE AREA
388	CITTFQ	PTR(31)	HEAD OF FREE TRACK HOLDERS
214	CITTR	FIXED(31)	ITTR PTR TO 1ST INP STRING
210	CITTRA	FIXED(31)	ALTERNATE ITTR PTR
38C	CITTWQ	PTR(31)	HEAD OF AVAIL. TRACK QUEUE
6E4	CJOBNAME	CHAR(8)	JOB NAME
4AA	CKEYMODE	PTR(8)	NUMBER OF INDEXES PER TRACK
398	CKEYSIZE	FIXED(31)	SIZE OF CONTROL WORD
3A8	CLASTIME	FIXED(31)	LAST ENTRY TO REAL TIME RTN
1EC	CLATCNT	FIXED(31)	# OF LA-INSTR. IN RCD-MOVE
3E8	CLCLTH	FIXED(31)	LENGTH LAST CLC. TREEGEN
3E4	CLCNUM	FIXED(31)	NO. OF CLC-INSTR. TREEGEN
7FC	CLENSTAT	PTR(31)	SMF WORK AREA PTR
41C	CLINK	FIXED(31)	ZERO IF NO MORE MOD TO CALL
778	CLINKREQ	CHAR(1)	FLAGS LINKEDITING REQUEST
7A8	CLISTMAD	CHAR(1)	TEMP SAVE FOR CLISTMOD
335	CLISTMOD	CHAR(1)	Y = LIST CONTROL CARDS
3AC	CLOCKS		TIME ACCUMULATORS
3AC	CLOCK0	FIXED(31)	C-PHASE ELAPSED TIME
3B0	CLOCK1	FIXED(31)	C-PHASE CPU TIME
3D4	CLOCK10	FIXED(31)	SPECIAL CLOCK 2
3D8	CLOCK11	FIXED(31)	SPECIAL CLOCK 3
3DC	CLOCK12	FIXED(31)	SPECIAL CLOCK 4
3E0	CLOCK13	FIXED(31)	SPECIAL CLOCK 5
3B4	CLOCK2	FIXED(31)	P-PHASE ELAPSED TIME
3B8	CLOCK3	FIXED(31)	P-PHASE CPU TIME
3BC	CLOCK4	FIXED(31)	R-PHASE ELAPSED TIME

DISP	LABEL	ATTR	COMMENT
3C0	CLOCK5	FIXED(31)	R-PHASE CPU TIME
3C4	CLOCK6	FIXED(31)	E-PHASE ELAPSED TIME
3C8	CLOCK7	FIXED(31)	E-PHASE CPU TIME
3CC	CLOCK8	FIXED(31)	SPECIAL CLOCK 0
3D0	CLOCK9	FIXED(31)	SPECIAL CLOCK 1
39C	CLOCSIZE	BDY(WORD)	TABLE FOR VARIABLE GETMAIN
779	CLODCNT1	CHAR(1)	LOAD COUNT PHASE 1
77A	CLODCNT2	CHAR(1)	LOAD COUNT PHASE 2
77B	CLODCNT3	CHAR(1)	LOAD COUNT PHASE 3
3E6	CLTHIN	FIXED(15)	INPUT RECORD LENGTH
408	CMAINSIZ	FIXED(31)	SIZE OF WORKING STORAGE
31C	CMANMSG	FIXED(31)	PTR TO MSGTABLE FOR RESTORE
324	CMAXLIM	FIXED(31)	MAX LIMIT FOR SIZE(MAX)
60C	CMAXREC	FIXED(31)	MAXIMUM RECORD SIZE
404	CMAXSIZE	FIXED(31)	MAXIMUM SIZE WANTED
3F4	CMERGE	PTR(31)	ADDR. OF TRMERGE ROUTINE
5E5	CMERGEIN	BIT(1)	SORTIN01 DD STMNT PRESENT
6DC	CMESSAGE	CHAR(120)	MESSAGE AREA
328	CMINLIM	FIXED(31)	MIN VIRTUAL STRG FOR PEER
400	CMINMAX	BDY(WORD)	PARMS FOR VARIABLE GETMAIN
40C	CMINREC	FIXED(31)	MINIMAL RECORD SIZE
40E	CMINRECH	FIXED(15)	HALF-WORD
400	CMINSIZE	FIXED(31)	MINIMUM SIZE ACCEPTED
4A6	CMODCARD	CHAR(1)	YES = MODS STMNT PRESENT
7B0	CMODDCB	FIXED(31)	MODSLIB DCB ADDRESS
410	CMODEVAL	FIXED(31)	MODAL RECORD SIZE
41F	CMODNAME	CHAR(3)	SIGNIFICANT PART OF MODNAME
418	CMODSEND	PTR(31)	ENDING BYTE
448	CMODSLIB	CHAR(8)	MODS LIBRARY NAME
414	CMODSTM	BDY(WORD)	MODS STATEMENT -
414	CMODSTRT	PTR(31)	STARTING BYTE
41C	CMODULE	CHAR(8)	MOD TO BE CALLED
412	CMODVALH	FIXED(15)	HALF-WORD
3FC	CMORETRK	PTR(31)	TRKS NOT TO RELEASE
430	CMOVEBR (3)	PTR(16)	CCD MVC BUFFER TO RSA
436	CMOVEBRI(3)	PTR(16)	INDEX MVC BUFFER TO RSA
178	CMOVEEND	PTR(31)	WORK AREA FOR MOVECOMP
442	CMOVEE35(3)	PTR(16)	MVC FOR USER RTN AT E35
17C	CMOVEH	PTR(31)	WORK AREA FOR MOVECOMP
3EC	CMOVEKEY(3)	PTR(16)	CTL WORD MVC OR BAL TO RTN
454	CMOVELNG	FIXED(31)	BYTES TO MOVE TO RSA
424	CMOVERB (3)	PTR(16)	RCD MVC RSA TO BUFFER
42A	CMOVERBC(3)	PTR(16)	RCD MVC RSA TO BUFFER(C-PH)
43C	CMOVERBI(3)	PTR(16)	INDEX MVC RSA TO BUFFER
7FA	CMSGACNT	FIXED(15)	COUNT OF ACTION MESSAGES
160	CMSGADDR	PTR(31)	ADDR. OF ROUTINE DIAGNOSE
6F5	CMSGBLNK	CHAR(1)	MESSAGE BLANK
6F6	CMSGINFO	CHAR(94)	MESSAGE TEXT
4F4	CMSGLIST (29)	FIXED(31)	MSG VARIABLE INFO LIST
6DF	CMSGNO	CHAR(3)	MESSAGE NUMBER
330	CMSGOPTN	CHAR(4)	MESSAGE OPTION
450	CMSGTABL	PTR(31)	ADDR. ERROR MESSAGE TABLE
6E2	CMSGTAG	CHAR(1)	MESSAGE TAG
3F8	CMSGTYPE	PTR(31)	DEBUG MESSAGE TYPE INDEX
7E3	CMULTVOL	CHAR(1)	MULTIVOLUME DATA SET FLAG
494	CNEWPAGE	BIT(1)	NEW PAGE HAS BEEN TAKEN
45C	CNEXTWRK	FIXED(31)	PRIM WORKD CHAIN
496	CNOREUSE	BIT(1)	VSAM OUTPUT NON-REUSABLE
458	CNUMWORK	FIXED(31)	NUMBER OF WORK AREAS
7E8	COBSBLOK	PTR(31)	ADDRESS OF EXCP DATA CODE
7D8	COBSEXIN	PTR(31)	ADDRESS OF SORTIN OPEN EXIT
7E4	COBSEXIT	PTR(31)	EXCP OPEN EXIT ADDRESS
7DC	COBSEXUT	PTR(31)	ADDRESS OF SORTOUT -//-
460	COBSPACE	FIXED(31)	OUTPUT BFR SPACE
588	COBUFFER	PTR(31)	ADDR OF OUTPUT INDEX BFR

DISP	LABEL	ATTR	COMMENT
39C	COMLOC	PTR(31)	GETMAINED CHAIN
174	COMLOCSV	FIXED(31)	COREPARM SAVE FOR ICEDEV
4B0	COMMB	PTR(31)	EXTRACT/RESTORE CODE AREA
4B4	COMMBEND	PTR(31)	END OF EXTRACT/RESTORE AREA
178	COMMDBLE	BDY(DWORD)	DOUBLEWORD-ALIGNED WORKAREA
3A0	COMSIZE	FIXED(31)	SIZE OF RETURNED STORAGE
38C	CONBLKSZ	FIXED(15)	BLKSIZE OF 1-ST SORTIN FILE
496	CONCHAIN	BIT(1)	NO CHAIN SCHED FOR SORTOUT
4B8	CONLRECL	FIXED(15)	
497	CONRECFM	CHAR(1)	
494	CONSW	PTR(8)	MISCELLANEOUS SWITCHES
495	CONSW1	PTR(8)	MISCELLANEOUS SWITCHES
496	CONSW2	PTR(8)	MISCELLANEOUS SWITCHES
5E5	CONSW3	PTR(8)	MISCELLANEOUS SWITCHES
496	COPENCLS	BIT(1)	OP/CL SYSOUT IN PRSS
7F9	COPENFL	CHAR(1)	PERFORM OPEN IN ICEOBS
464	COPSRTIN	PTR(31)	OPEN PTR USED IN DEF + CRE
498	COPTIONS		OPTION SWITCHES
320	COREPARM	FIXED(31)	MAXIMUM ALLOWED
321	COREPM24	PTR(24)	FOR SORT/MERGE
5B0	COREUSED	FIXED(31)	CORE USED FOR MSG 92/93
5A0	CORMSG39	PTR(31)	CORE VALUE FOR MSG 39 (DEV)
470	COUNTERS		EVENT COUNTERS
470	COUNTER0	FIXED(31)	ENTRIES TO INOUTRTN
474	COUNTER1	FIXED(31)	ISSUED EXCPS
478	COUNTER2	FIXED(31)	RECORDS IN
47C	COUNTER3	FIXED(31)	RECORDS OUT
480	COUNTER4	FIXED(31)	RECORDS INSERTED
484	COUNTER5	FIXED(31)	RECORDS DELETED
488	COUNTER6	FIXED(31)	SPECIAL COUNT 0
48C	COUNTER7	FIXED(31)	SPECIAL COUNT 1
490	COUNTERX	FIXED(31)	TEMPORARY COUNT 1
7A4	COUNTERY	FIXED(31)	RECS DELETED BY E25
46C	COUTLIM	PTR(31)	ADDR. OF END OF WORK BUF
468	COUTSIZE	FIXED(31)	SORTOUT LOGICAL RCD LENGTH
4D0	CPAGELIM	PTR(31)	END PTR TO UNPROT FIXD AREA
4D4	CPAGELOC	PTR(31)	ADDR. OF FIXED AREAS TABLE
5A0	CPAGERSA	PTR(31)	PTR TO END OF LAST BUFFD0
4AC	CPAGESIZ	PTR(31)	PAGE SIZE FOR SYSTEM
4DC	CPARCLC (3)	PTR(16)	PAR INDEX BIN COMP
495	CPARMODE	BIT(1)	PARTITION MODE SWITCH
4E2	CPARMVC (3)	PTR(16)	PAR INDEX BIN MOVE
	CPATCH	FIXED(31)	RESERVED
5A4	CPBUF1	FIXED(31)	INDEX POOL ADDR
4D8	CPEERAGE	FIXED(31)	MIN. NO STRINGS ON SORTWK
56C	CPHASECG	FIXED(31)	MAXIMUM G-VALUE - CREATION
578	CPHASEEG	FIXED(31)	MAXIMUM G-VALUE-ELIMINATION
570	CPHASEPG	FIXED(31)	MAXIMUM G-VALUE - PARTTION
574	CPHASERG	FIXED(31)	MAXIMUM G-VALUE - REDUCTION
57C	CPHASEWK	BDY(WORD)	PHASE WORK AREA
7A9	CPHASLNK	CHAR(1)	TO REMEMBER THAT WE LINK
56B	CPREGION	CHAR(1)	REGION IS CORE LIMIT
332	CPRINMOD	CHAR(1)	Y = PRINT ALL MESSAGES
5AC	CPRMSIZ	FIXED(31)	PERM ALLOC SIZE
5A8	CPROTSIZ	FIXED(31)	PROT AREA SIZE
4EA	CPRT	CHAR(129)	(WORD 3) PRINT BUFFER
4BC	CPRTDECB(5)	PTR(31)	PRINTER OUTPUT DECB
4EA	CPRTHEAD (4)	FIXED(15)	VLR AND VLB INFORMATION
4F2	CPRTLINE	CHAR(121)	(WORD 3) PRINTER OUTPUT
331	CPRTRMOD	CHAR(1)	Y = MESSAGES TO PRINTER
59C	CPUTREC	PTR(31)	ADDR. OF OUTPUT PUT ROUTINE
4BA	CPOSIZE	FIXED(15)	PARTITION ZERO SIZE
14C	CRALADDR	PTR(31)	ADDR. OF ROUTINE RALLOCAT
5D4	CRBUFFRS	FIXED(16)	NUMBER OF R-PHASE BUFFERS
5B0	CREBUILD	PTR(31)	REBUILD SPANNED RECORDS

DISP	LABEL	ATTR	COMMENT
5B8	CRECADDR	PTR(31)	ADDR. OF RECORD - FOR VSAM
5BC	CRECAREA	PTR(31)	VLR PAD/RESTORE AREA
5C0	CRECCTR	FIXED(31)	RCD COUNT FOR CURRENT TRACK
3E8	CRECFMIN	CHAR(1)	INPUT RECORD FORMAT
168	CRECHAIN	PTR(31)	ADDR. OF ROUTINE RECHAIN
5C4	CRECORD	FIXED(31)	ADDR TO RECORD KEY
5CC	CRECPBUF	FIXED(31)	RECORDS/BUFFER
5C8	CRECPTR	PTR(31)	SPANNED REC PTR
5B4	CRECSIZE	FIXED(31)	SIZE OF TRANSPOSED RECORD
5D0	CRECTBUF	FIXED(31)	RCD/BUF *TREE ENTRY SIZE(4)
5D7	CRECTYPE	PTR(8)	RECORD TYPE
343	CRELSE	CHAR(1)	Y=RELEASE WORKSPACE
342	CRESDNT	CHAR(1)	A=ALL M=MAN N=NONE
32C	CRESERVE	FIXED(31)	STORAGE
5D8	CRESPMTM	PTR(31)	PERM RES CH OR TEMP RES
5E0	CRESTMP	PTR(31)	TEMP RESERVE CNT
5DC	CRESTORE	PTR(31)	ADDR. OF RESTORE ROUTINE
5D6	CRETCODE	PTR(8)	RETURN CODE AREA (0|10)
5E4	CRLSE	CHAR(1)	YES FOR RELEASE OF SORTWK
5E8	CRMERGE	FIXED(31)	MERGE ORDER - REDUCTION
5EC	CRPLIN	PTR(31)	ADDR OF VSAM INPUT RPL
5F0	CRPLOUT	PTR(31)	ADDR OF VSAM OUTPUT RPL
5F4	CRSAFREE	PTR(31)	ADDR. OF NEXT BIN IN RSA
6D4	CRSAFULL	BIT(1)	1=RSA IS FULL
32D	CRSRVD24	PTR(24)	CALLING PROGRAM
140	CRTNADDR		ADDR. OF PERMANENT RTNS
048	CSAVEL1		LEVEL 1 ROUTINE SAVE AREA
090	CSAVEL2	PTR(31)	LEVEL 2 ROUTINE SAVE AREA
0D8	CSAVEL3		LEVEL 3 ROUTINE SAVE AREA
000	CSAVEOS	PTR(31)	SUPERVISOR AND DM SAVE AREA
690	CSAVEPTR	FIXED(31)	SAVE AREA POINTER
7F4	CSAVER14	FIXED(31)	SAVE AREA IN ALL PHASES
610	CSAVITTR	PTR(31)	PREVIOUS INDEX BLOCK ITTR
388	CSAVREG7	FIXED(31)	SAVE INPUT PTR (ICELIM)
60C	CSCANLOC	PTR(31)	ADDR. OF LAST SCAN LOCATION
344	CSECALC	CHAR(1)	Y=AUTOM. SEC. ALLOC.
684	CSECALL	CHAR(2)	SECONDARY ALLOCATION SWITCH
68C	CSECTRK	FIXED(31)	SEC.ALLOC. TRK COUNTER
4A2	CSEEKMOD	CHAR(1)	YES = DO STAND-ALONE SEEKS
618	CSELECT	PTR(31)	ADDR. OF TSELECT ROUTINE
494	CSFROMSW	BIT(1)	FROM-REG IN RCD-MOVE SAVED
495	CSHORTSW	BIT(1)	SHORT VAR RECORD FOUND
494	CSKIP	BIT(1)	SKIP PHASES CONTROL SWITCH
608	CSKIPREC	FIXED(31)	NUMBER OF RECORDS TO SKIP
609	CSKIPR24	PTR(24)	THREE BYTES
34B	CSMF	CHAR(1)	S=SHORT SMF RECORD
600	CSORTIN	FIXED(31)	SORTIN DCB POINTER
498	CSORTNAM	CHAR(4)	PREFIX FOR DD STATEMENTS
604	CSORTOUT	FIXED(31)	SORTOUT DCB POINTER
61C	CSORTWK	PTR(31)	PTR TO SORTWK DCB LIST
5D7	CSPANIN	BIT(1)	ON = VAR. SPANNED INPUT
5B8	CSPNVSAM	PTR(31)	
5E5	CSRTCNTL	BIT(1)	SORTCNTL STATEMENT PRESENT
774	CSRTMODS	PTR(31)	SORTMODS DCB POINTER
49C	CSRTMRGE	CHAR(1)	SORT=(S) | MERGE=(M)
16C	CSTAADDR	PTR(31)	ADDR OF INIT STAE
34D	CSTAE	CHAR(1)	Y = STAE WANTED AT ABEND
688	CSTAESAV	PTR(31)	STAE WORK AREA -
689	CSTAES24	PTR(24)	FOR NEXT LEVEL
6ED	CSTEPNAM	CHAR(8)	STEP NAME
7E0	CSTOPIN	CHAR(1)	CANT USE EXCP FOR SORTIN
7E1	CSTOPOUT	CHAR(1)	CANT USE EXCP FOR SORTOUT
494	CSTOSW	BIT(1)	TO-REG IN RCD-MOVE SAVED
614	CSTRINGS	FIXED(31)	STRING COUNT
798	CSTWNAME	CHAR(12)	FOR STOW MACRO

DISP	LABEL	ATTR	COMMENT
682	CSUBPBAS	FIXED(15)	SUB-BIN/BASE-BIN
680	CSUBSIZE	FIXED(15)	SUB-BIN SIZE
348	CSVC	PTR(16)	SVC INSTRUCTION
49D	CSYSDUMP	CHAR(1)	YES = SYSUDUMP OR SYSABEND
5F8	CSYSIN	PTR(31)	SYSIN DCB POINTER
774	CSYSLIN	PTR(31)	SYSLIN DCB POINTER
7B4	CSYSLMOD	FIXED(31)	SYSLMOD DCB ADDRESS
338	CSYSNAME	CHAR(8)	NAME FOR PRINT DD CARD
5FC	CSYSOUT	PTR(31)	SYSOUT DCB POINTER
620	CSYSOUTD	PTR(31)	SYSOUT - DCB
694	CSYSRES	FIXED(31)	SYSTEM NEEDS
4AB	CSYSTEM	CHAR(1)	O=MFT/MVT S=SVS M=MVS
788	CTABPTR	PTR(31)	POINTER IN SCANTAB
494	CTECHNQS	BIT(1)	TECHNIQUE FORCED
4A9	CTECHNQ	CHAR(1)	P|B|O|X|N OR DEFAULT E
120	CTEMP1	FIXED(31)	WORK AREA
123	CTEMP108	PTR(8)	WORK AREA
122	CTEMP115	FIXED(15)	WORK AREA
121	CTEMP124	PTR(24)	WORK AREA
124	CTEMP2	FIXED(31)	WORK AREA
127	CTEMP208	PTR(8)	WORK AREA
126	CTEMP215	FIXED(15)	WORK AREA
125	CTEMP224	PTR(24)	WORK AREA
128	CTEMP3	FIXED(31)	WORK AREA
12B	CTEMP308	PTR(8)	WORK AREA
12A	CTEMP315	FIXED(15)	WORK AREA
129	CTEMP324	PTR(24)	WORK AREA
12C	CTEMP4	FIXED(31)	WORK AREA
12F	CTEMP408	PTR(8)	WORK AREA
12E	CTEMP415	FIXED(15)	WORK AREA
12D	CTEMP424	PTR(24)	WORK AREA
6B0	CTESTKEY(3)	PTR(16)	CLC FOR TWO CONTROL WORDS
1D0	CTFCB	FIXED(31)	TFCB CHAIN HEAD
164	CTIME	PTR(31)	ADDR. OF ROUTINE TIMING
358	CTIOTIN	PTR(31)	PTR TO INPUT DD NAME (DEF)
59C	CTIOTOUT	PTR(31)	PTR OUTPT DD NAME (DEF-DEG)
6A8	CTMPSIZ	FIXED(31)	TEMP ALLOC SIZE
808	CTOTTIME	FIXED(31)	CPU TIME AREA FOR SMF
495	CTPDYNSW	BIT(1)	NO DYNNUMB FOR TAPE
828	CTRACE	CHAR(128)	TRACE TABLE
810	CTREEBAS	FIXED(31)	TREE BASECODE ADDRESS
69C	CTREELOC	PTR(31)	ADDR. OF START OF TREE
698	CTREESIZ	FIXED(31)	SIZE OF ALLOCATED TREE
6AC	CTREE2	PTR(31)	LOOK AHEAD TREE
6A4	CTRESHLD	FIXED(31)	FREE BYTES IN RSA
158	CTRKROUT	PTR(31)	ADDR. OF ROUTINE TRKROUT
6BC	CURRDATE	FIXED(31)	CURRENT DATE - PACKED DEC.
6B8	CURRDECB	PTR(31)	ADDR. OF CURRENT DECB
6C0	CURREC	PTR(31)	ADDR TO BUFFERED REC IN RSA
6A0	CUTABENT	PTR(31)	PTR/DISPL TO UNIT TAB ENTRY
5D7	CVARREC	BIT(1)	ON = VARIABLE LENGTH RECS
34C	CVBLKSET	CHAR(1)	N =BYPASS VLR-BLOCKSET
6C4	CVERACC	FIXED(31)	ACCUMULATOR FOR VERIFY
345	CVERIFY	CHAR(1)	Y=VERIFY OUTPUT RECS
6D4	CVIM	BIT(1)	1=BYPASS VED CALL VIM
340	CVIO	CHAR(1)	VIO UNDER MVS
34E	CVIOREC	CHAR(1)	VIO OR TAPE REC
6D0	CVMOVEL	FIXED(31)	VLR - NO OF BYTES TO MOVE
6D4	CVREMSW	BIT(8)	ICEVRE MISC SWITCH
6C8	CVSAMERR	PTR(31)	ADDR OF MODULE ICEERR
6CC	CVSAMRVE	FIXED(31)	STORAGE RESEVED FOR VSAM
496	CVSAMR1	BIT(1)	VSAM RELEASE 1 IN SYSTEM
75C	CWBPOOL	FIXED(31)	WB-POOL ADDRESS
760	CWBPOOLL	FIXED(15)	WRITEBACK POOL SIZE
762	CWBPOOLC	FIXED(15)	CURRENT POOL SIZE

DISP	LABEL	ATTR	COMMENT
764	CWBPOOLR	FIXED(15)	OFFSET TO RESERVE BLOCK
5E5	CWKVAMDS	BIT(1)	VIO WORK DATASET PRESENT
6D6	CWMINCON	FIXED(15)	CTRL FIELD BASED MIN RECLEN
766	CWMINREC	FIXED(15)	MIN EXTRACT IN NON-PEER
758	CWORKQ	PTR(31)	ADDRESS OF WORK AREA QUEUE
120	CWORK1	FIXED(31)	WORK AREA
123	CWORK108	PTR(8)	WORK AREA
122	CWORK116	FIXED(16)	WORK AREA
121	CWORK124	PTR(24)	WORK AREA
124	CWORK2	FIXED(31)	WORK AREA
127	CWORK208	PTR(8)	WORK AREA
126	CWORK216	FIXED(16)	WORK AREA
125	CWORK224	PTR(24)	WORK AREA
128	CWORK3	FIXED(31)	WORK AREA
12B	CWORK308	PTR(8)	WORK AREA
12A	CWORK316	FIXED(16)	WORK AREA
129	CWORK324	PTR(24)	WORK AREA
12C	CWORK4	FIXED(31)	WORK AREA
12F	CWORK408	PTR(8)	WORK AREA
12E	CWORK416	FIXED(16)	WORK AREA
12D	CWORK424	PTR(24)	WORK AREA
130	CWORK5	FIXED(31)	WORK AREA
134	CWORK6	FIXED(31)	WORK AREA
138	CWORK7	FIXED(31)	WORK AREA
13C	CWORK8	FIXED(31)	WORK AREA
6D5	CWRKTYPE	CHAR(1)	T=TAPE D=DISK
6D8	CWTO	BDY(WORD)	WTO - MF=L
178	CXCCOUNT	FIXED(31)	WORK AREA FOR EXTRACT
17C	CXCLNG	FIXED(31)	WORK AREA FOR EXTRACT
170	CXLIST	PTR(31)	ADDRESS OF EXIT LIST
800	CXTRSTAT	PTR(31)	SMF STATISTICS RTN PTR
7B8	CZONE61	CHAR(32)	INTERM. STOR. F. ZD W. E61

Appendix A: Program Exits

Calling Modules

	Disk								Tape								Merge
Exit	BLOC FLR	BLOC VLR	PEER	VALE	CRCX fix	CRCX var	BALN fix	BALN var	OSCL fix	OSCL var	POLY fix	POLY var	BALN fix	BALN var	-only
Ph0															
E18	-	-	DED	DED	-	-	-	-	-	-	-	-	-	-	-
E39	-	-	DED	DED	-	-	-	-	-	-	-	-	-	-	-
Ph1															
E11	-	-	-	VRE[1]	APL	APL	APG	APG	APL	APL	APG	APG	APG	APG	-
E15	E15[2]	15V[8]	CRE	VRE[1]	RDR	RDS	RDD	RDE	RDR	RDS	RDD	RDE	RDD	RDE	-
E16	-	-	-	VRE[1]	RDR	RDS	RDD	RDE	RDR	RDS	RDD	RDE	RDD	RDE	-
E17	-	-	-	VRE[3]	8PM	8PM	RPC	RPC	RPM	RPM	RPC	RPC	RPC	RPC	-
E18	-	-	-	VRE[3]	9GN	9GN	AGI	AGI	AGN	AGN	AGA	AGA	AGA	AGA	-
E19	-	-	-	VRE	9GN	9GN	AGI	AGI	AGN	AGN	AGA	AGA	AGA	AGA	-
E31	-	-	-	VRO	-	-	-	-	-	-	-	-	-	-	-
E35	-	-	CRO	VRE	-	-	-	-	-	-	-	-	-	-	-
E37	-	-	-	VRO	-	-	-	-	-	-	-	-	-	-	-
E39	-	-	-	VRO	-	-	-	-	-	-	-	-	-	-	-
E61	-	-	CRE	VRE[1]	ROB	ROD	ROB	ROD	ROB	ROD	ROF	ROH	ROB	ROD	-
Ph2															
E31	-	-	-	VED	APL	APL	APH	APH	APL	APL	APH	APH	APH	APH	-
E25	-	-	-	VEE	RBW	RBX	RBJ	RBK	RBW	RBX	RBJ	RBK	RBJ	RBK	-
E27	-	-	-	VED	8PM	8PM	RPF	RPF	RPM	RPM	RPF	RPF	RPF	RPF	-
E28	-	-	-	-	9GN	9GN	AGJ	AGJ	AGN	AGN	AGG	AGG	AGG	AGG	-
E29	-	-	-	-	9GN	9GN	AGJ	AGJ	AGN	AGN	AGG	AGG	AGG	AGG	-
E61	-	-	-	VEE	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	-
Ph3	-	-													
E31	-	-	-	VIP	API	API	API	API	API	API	API	API	API	API	API
E32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RDU
E35	E35[4]	35VL	LIM[5]	VIM	RBM	RBO[6]	RBM	RBO[6]	RBM	RBO[6]	RBM	RBO[6]	RBM	RBO[6]	RBM[7]
E37	-	-	-	VIM	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
E38	-	-	-	-	AGK	AGK	AGK	AGK	APK	APK	APK	APK	APK	APK	APF
E39	-	-	LIM	VIP	AGK	AGK	AGK	AGK	APK	APK	APK	APK	APK	APK	APF
E61	-	-	-	-	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ

[1]Or VRN
[2]Or IPUM if no SORTIN
[3]Or VRN or VRO
[4]Or COBU if no SORTOUT
[5]Of LIV
[6]8BO for spanned records
[7]RBO for variable-length records
[8]Or COBV if no SORTIN

Register Usage

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

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

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

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

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

Appendix B: Format Codes (Conventional Techniques)

With a conventional sort or merge each node in the tree contains a code describing the current status of the node.

The structure of a node, and the various format codes as they appear in the second word of that node, are described below.

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

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

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

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

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

Fixed-Length Records

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

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

Variable-Length Records

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

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

FORMAT CONDITION CODES

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

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

Example: Fixed-Length Records

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

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

Example: Variable-Length Records

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

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

Appendix C. Checkpoint/Restart Facility

If it is included at system generation the sort/merge program may make use of the Checkpoint/Restart Facility of the operating system. The user directs the sort/merge program to use this facility by (1) including the checkpoint parameter (CKPT) in the SORT or MERGE control statement, and (2) providing a SORTCKPT DD statement to define the checkpoint data set; this data set will store the information needed to restart processing.

Checkpoint procedures are described below for Peerage and Vale, and for the conventional sort/merge techniques.

With Blockset checkpoints are not taken. If a checkpoint has been requested the Blockset technique is rejected and another used instead.

Sorting Application (Peerage and Vale)

If checkpoints are requested, the checkpoint macro (CHKPT) is issued and checkpoint records are written on the checkpoint data set as follows:

- ICECKP/ICEVED start of intermediate merge phase
- ICEVIM start of final merge phase
- when the checkpoint track pool is empty.

The relative track addresses of 20-30% (maximum 1364 tracks) of the tracks primarily allocated to the intermediate work files are saved in a track pool. Every time a track of blocks has been read, the relative address is saved in the track pool. When the blocks have to be written the address of an unused track is found in the track pool. A checkpoint is taken when there are no more unused tracks. After the checkpoint has been taken, the saved tracks are available for use again.

Sorting Application (Conventional Techniques)

If checkpoints are requested the following modules link to module ICECHK. This issues checkpoint macro instruction (CHKPT), which causes checkpoint records to be written on the checkpoint data set.

- ICEAPC Start of phase 1 (all techniques)
- ICEAPJ Start of phase 2 (all techniques except crisscross and oscillating)
  Start of each intermediate merge phase pass (balanced direct access technique)
- ICERON During phase 2 (oscillating technique)
- ICEROS During phase 2 (polyphase technique)
- ICEROT During phase 2 (balanced DASD technique)
- ICEAGH Start of phase 3 (all techniques)

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

Merge-Only Application

If checkpoints are requested they are taken whenever an end-of-volume condition occurs on SORTOUT, unless the user provides his own exit list address for the SORTOUT DCB. Module ICEAPF inserts the address of the checkpoint module ICE8CR in the DCBEXLST field when it generates the SORTOUT DCB (unless this field has been set by a user routine at exit E39). Module ICE8CR is thus entered at the DCB exit list, which contains only the end-of-volume entry. This entry points to the EOV exit routine in this module, which in turn issues a checkpoint macro instruction (CHKPT), and then returns to the IOCS.

Appendix D. Program Listing Standards and Conventions

To facilitate the identification of modules, work areas, tables, and other aspects of the sort/merge program listing, each technique assigns symbolic names according to a pattern.

Blockset

MODULE NAMES

The format of all module names except ICEMAN and ICEAM1 is ICEmod, where:

ICE is the identification for program modules

mod is a descriptive name for the module (four characters)

INSTRUCTION NAMES

Descriptive names are used. All variables, which are resident in COMMON, begin with the characters 'COM'.

CONSTANT NAMES

Descriptive names are used. 'Constant' is used to mean any item of data defined outside COMMON which is not dynamic (i.e. based on a pointer).

WORK AREA NAMES

Since the program is reenterable, all work areas are variables, and are thus resident in COMMON

Peerage and Vale

MODULE NAMES

The format of all module names is ICEmod, as described under 'Blockset' above. The descriptive portion (mod) is 3 characters long.

INSTRUCTION NAMES

Descriptive names are used. All variables, which are resident in ICECOMMA, begin with the character 'C'.

CONSTANT NAMES

Descriptive names are used. 'Constant' is used to mean any item of data defined outside ICECOMMA which is not dynamic (i.e. based on a pointer).

WORK AREA NAMES

Since the program is reenterable, all work areas are variables, and are thus resident in ICECOMMA.

Conventional Techniques

MODULE NAMES

The format of module names is ICEtmm, where:

ICE is the identification for program modules.

t, in general, is either an 'A' for an Assignment or an 'R' for a Running type module; however, for some assignment modules associated with the crisscross technique, t is '9', and for some running modules associated with the crisscross technique, t is '8'.

mm is the unique portion of the module name.

INSTRUCTION NAMES

The format of all internal type instruction names is mmnnnnno or mmmnnnno where:

mm or mmm represents the last two or three characters of the module name.

nnnnn or nnnn is a unique designation assigned by the programmer and may be from one to five characters.

o is an 'X' if the label is externally used; otherwise it can be used as another n.

Appendix E: SMF Record DSECT (ICESMF)

This area is allocated and filled in by ICEEXIO, ICEOXOV, or ICEMOS depending on the sort technique used. The record is passed to sort's SVC routine, ICEFIXM (OS/VS2) or ICEFIX (OS/VS1). The SVC routine uses the passed record and issues the SMF macro to write the SMF record out to the SMF data set, upon successful completion of the sort/merge application.

```
DISP  NAME                    DESCRIPTION
0000   ICESMF   DSECT
0000   ICERDW   DS    XL4     RECORD DESCRIPTOR WORD
0004   ICESIND  DS    B       SYSTEM INDICATOR
         *                      BIT0: SUBSYSTEM ID FOLLOWS SYSTEM ID
0005   ICERTYP  DS    X       SMF RECORD TYPE. (16)
0006   ICEBTIME DS    FL4     TIME RECORD WAS MOVED TO SMF BUFFER
000A   ICEBDATE DS    PL4     DATE RECORD WAS MOVED TO SMF BUFFER
000E   ICESID   DS    CL4     SYSTEM IDENTIFICATION
0012   ICEJOBNM DS    CL8     JOBNAME
001A   ICERST   DS    FL4     TIME READER RECOGNIZED JOBCARD
001E   ICERDS   DS    PL4     DATE READER RECOGNIZED JOBCARD
0022   ICEUIF   DS    CL8     USER ID (TAKEN FROM COMMON EXIT
         *                      PARAMETER AREA)
002A   ICESTN   DS    XL1     STEP NUMBER
002B   ICERES1  DS    CL3     RESERVED
002E   ICESUBID DS    CL4     SUBSYSTEM ID
0032   ICERSUB  DS    XL2     RECORD SUBTYPE
         ICERSUBS EQU   X'01'  SHORT RECORD
         ICERSUBF EQU   X'02'  FULL RECORD
         ********************************************************************
         *  SELF DEFINING SECTION                                           *
         ********************************************************************
0034   ICEPROD  DS    F       OFFSET TO PRODUCT SECTION
0038   ICEPRODL DS    H       PRODUCT SECTION LENGTH (10)
003A   ICEPRODN DS    H       NUMBER OF PRODUCT SECTIONS (1)
         *
003C   ICEDATA  DS    F       OFFSET TO SECTION COMMON TO SHORT AND FULL RECS
0040   ICEDATAL DS    H       DATA SECTION LENGTH (46)
0042   ICEDATAN DS    H       NUMBER OF DATA SECTIONS (1)
         *
0044   ICESTAT  DS    F       OFFSET TO RECORD LENGTH STATISTICS
0048   ICESTATL DS    H       STATISTICS SECTION LENGTH (64)
004A   ICESTATN DS    H       NUMBER OF DATA SECTIONS (1|0)
         ********************************************************************
         *  PRODUCT SECTION                                                 *
         ********************************************************************
004C   ICERECV  DS    CL2     RECORD VERSION. '50' FOR RELEASE 5.0
004E   ICEPRDCT DS    CL8     PRODUCT NAME. '5740-SM1'
         ********************************************************************
         *  DATA SECTION. COMMON PART                                       *
         ********************************************************************
0056   ICECDAT  DS    0H      START OF DATA SECTION
0056   ICERES2  DS    CL2     TO PUT NEXT FIELD ON FULLWORD BOUNDRY
0058   ICESTPNM DS    CL8     STEPNAME
0060   ICERCDS  DS    F       NUMBER OF RECORDS SORTED
0064   ICEBYTES DS    F       NUMBER OF BYTES SORTED (SUM OF RECORD LENGTHS)
0068   ICECPUT  DS    F       SORT CPU TIME, HUNDREDTHS OF A SECOND
006C   ICELEN   DS    H       SPECIFIED RECORD LENGTH
006E   ICEIBLK  DS    H       INPUT BLOCKSIZE (MAX)
0070   ICEOBLK  DS    H       OUTPUT BLOCKSIZE
0072   ICEKEYLN DS    H       TOTAL CONTROL FIELD LENGTH (NUMBER
         *                      OF BYTES ACTUALLY COMPARED BY SORT)
```

```
DISP  NAME                    DESCRIPTION
0074   ICEWBLK  DS    F       NUMBER OF WORK DATA SET TRACKS USED
0078   ICEFLBYT DS    B       BIT 0: RESERVED
         *                       BIT 1-2: =FIXED LENGTH RECORDS
         *  01=VARIABLE LENGTH RECORDS
         *  10=VARIABLE LENGTH SPANNED RECORD
         *                       BIT 3-4: =BLOCKSET
         *  01=PEERAGE
         *  10=VALE
         *  11=CONVENTIONAL AND MERGE TECHNIQUES
         *                       BIT 5: '1'B IF SORT DYNAMICALLY INVOKED
         *                       BIT 6-7: RESERVED
0079   ICENDYNA DS    AL1     NUMBER OF DYNAMICALLY
         *                       ALLOCATED WORK DATA SETS
007A   ICERES3  DS    CL2     RESERVED
         ************************************************************
         *  DATA SECTION. STATISTICS PART                           *
         ************************************************************
         ICEVAR   EQU   ICECTR BEGINNING OF VARIABLE PART
007C   ICECTR   DS    16F     RECORD COUNTERS FOR INTERVALS 1 - 16
00BC           ORG   ICECTR
007C   ICECTR01 DS    F       RECORDS IN INTERVAL 1
0080   ICECTR02 DS    F       RECORDS IN INTERVAL 2
0084   ICECTR03 DS    F       RECORDS IN INTERVAL 3
0088   ICECTR04 DS    F       RECORDS IN INTERVAL 4
008C   ICECTR05 DS    F       RECORDS IN INTERVAL 5
0090   ICECTR06 DS    F       RECORDS IN INTERVAL 6
0094   ICECTR07 DS    F       RECORDS IN INTERVAL 7
0098   ICECTR08 DS    F       RECORDS IN INTERVAL 8
009C   ICECTR09 DS    F       RECORDS IN INTERVAL 9
00A0   ICECTR10 DS    F       RECORDS IN INTERVAL 10
00A4   ICECTR11 DS    F       RECORDS IN INTERVAL 11
00A8   ICECTR12 DS    F       RECORDS IN INTERVAL 12
00AC   ICECTR13 DS    F       RECORDS IN INTERVAL 13
00B0   ICECTR14 DS    F       RECORDS IN INTERVAL 14
00B4   ICECTR15 DS    F       RECORDS IN INTERVAL 15
00B8   ICECTR16 DS    F       RECORDS IN INTERVAL 16
         ICESMFND EQU   *       END OF RECORD
                  END
```

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

OS/VS Sort/Merge Logic

LY33-8042-6

READER'S COMMENT FORM

We would appreciate your comments. Please give specific page and line references where appropriate. Your comments will be carefully reviewed by the persons responsible for writing and publishing this material.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

If you wish a reply, please be sure to include your name and address.

COMMENTS

Reply requested:

Yes ☐

No ☐

Date

Name: . . . . . .

Job Title: . . . . . .

Address: . . . . . .

. . . . . Zip . . . . . .

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments, or you can send them direct: *IBM Corporation, P. O. Box 50020, Programming Publishing, San Jose, California 95150, U.S.A.*

THANK YOU FOR YOUR COOPERATION
PLEASE FOLD ON TWO LINES, STAPLE AND MAIL

Reader's Comment Form

Your comments about this publication will help us to produce better publication for your use. Each reply will be carefully reviewed by the person responsible for writing and publishing this material. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

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

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 825
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape | Please Do Not Staple | Fold and tape

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Cut or Fold Along Line

Technical Newsletter

This Newsletter No. LN20-9345
Date 31 March 1981

Base Publication No. LY33-8042-6
File No. S370-33 (OS/VS)

Prerequisite Newsletters LN20-9329

OS/VS Sort/Merge Logic

This technical newsletter, a part of Release 5 of Program Number 5740-SM1, provides replacement pages for the subject publication. These replacement pages remain in effect for any subsequent OS/VS releases unless specifically altered. Pages to be inserted and/or removed are:

Cover – ix
1 – 7 (2.1 and 2.2 added, 8 deleted)
17 – 20
23 – 32
57 – 60
67, 68
85 – 96 (86.1 and 94.1 added)
109, 110
121 – 138
157, 158
161, 162
171, 172
179 – 184
189 – 192
195, 196
203 – 218
228.1 – back cover (228.1 and 228.2 added)

Summary of Amendments

Release 5 of 5740-SM1 provides improved performance, 3375 direct access storage device support, and usability enhancements.

Note: *Please file this cover letter at the back of the publication to provide a record of change.*

IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150

Printed in U.S.A.

Technical Newsletter

This Newsletter No. LN20-9390
Date 29 April 1982

Base Publication No. LY33-8042-6
File No. S370-33 (OS/VS)

Prerequisite Newsletters LN20-9329 (obsolete)
LN20-9345

OS/VS Sort/Merge Logic

This technical newsletter, a part of Release 5, of OS/VS Sort/Merge, Program Product 5740-SM1, provides replacement pages for the subject publication. These replacement pages remain in effect for any subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

cover, edition notice
1-2.2

Each technical change is marked by a vertical bar to the left of the change.

Summary of Amendments

This newsletter contains information about the IBM 3880 Models 2, 3, and 13 Control Units.

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

IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150

Printed in U.S.A.

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601