IBM

Programming Systems Analysis Guide

7070/7074 Sort 90

IBM

Programming Systems Analysis Guide
7070/7074 Sort 90

Preface

This manual was prepared by Applied Programming to provide detailed information on the internal logic of the 7070/7074 Sort 90 Programming System. It is intended for technical personnel who are responsible for diagnosing the system operation or for adapting the programming system to special usage.

Certain knowledge is a prerequisite for the full utilization of this manual. It is assumed that the reader is familiar with Sort 90 to the extent that it is described in the *IBM 7070/7074 Sort 90* Operation Bulletin (J28-6096). In addition, it is assumed that the reader has an understanding of the IBM 7070 Input-Output Control System, major portions of which are an integral part of Sort 90. Such background can be obtained from the *IBM 7070 Input-Output Control System* Bulletin (J28-6033) and the *IBM Programming Systems Analysis Guide, 7070 Input-Output Control System* (C28-6119).

A program listing may be obtained by sending a 2,400 foot tape to the 7070 Program Library at Data Processing Library Services, 590 Madison Avenue, New York 22, New York.

Contents

The over-all goal of Sort 90 is to produce an output file of records in sequence with respect to specified control data, given a file randomly ordered (or, at least, not completely sequenced) with respect to those data. Sort 90 includes three phases (Chart 1A): Phase I reads the input file, forms successive groups of records into sequenced form by an internal sorting procedure, and writes the sequences on work tapes. Phase II consists of several passes of a symmetrical merging operation; in each pass, the sequences produced by Phase I or by the previous Phase II pass are merged together to reduce the total number of sequences, increasing the length of each. When a Phase II pass produces a small enough number of sequences, Phase III is entered; it performs the final merging pass, producing an output file which the customer may use as input for another program and/or as input for off-line operations.

The flow charts for each phase follow the description of that phase.

Sort 90 is a "generalized" program. This means that the program, immediately after being loaded, cannot perform any sorting operation whatsoever; the program must first analyze the control cards prepared by the user and modify itself so as to be capable of performing the specific sorting application desired. Each phase is therefore divided into two sections — a "running" program, which actually performs the sorting, and an "assignment" program, which sets up the running program and is then destroyed by record read-in.

An important feature of Sort 90 is that extensive provision is made for added programming, such as editing and summarizing. Thus, sorting may be only one of several major functions being performed during a run of this program. The sort may be inserted on a program tape along with other programs to form a link in an integrated system of programs.

A companion to Sort 90 exists in the generalized Merge 91 program, which may be used to merge or sequence-check already ordered files. Specifications for acceptable input record and file configurations, amount and type of control data, and collating sequences are identical for these two programs, and they may be used together. For example, should a file contain too many records for one run of Sort 90, it may be broken into subfiles. Each subfile is ordered by Sort 90, and the ordered subfiles are merged into a single sequence with Merge 91.

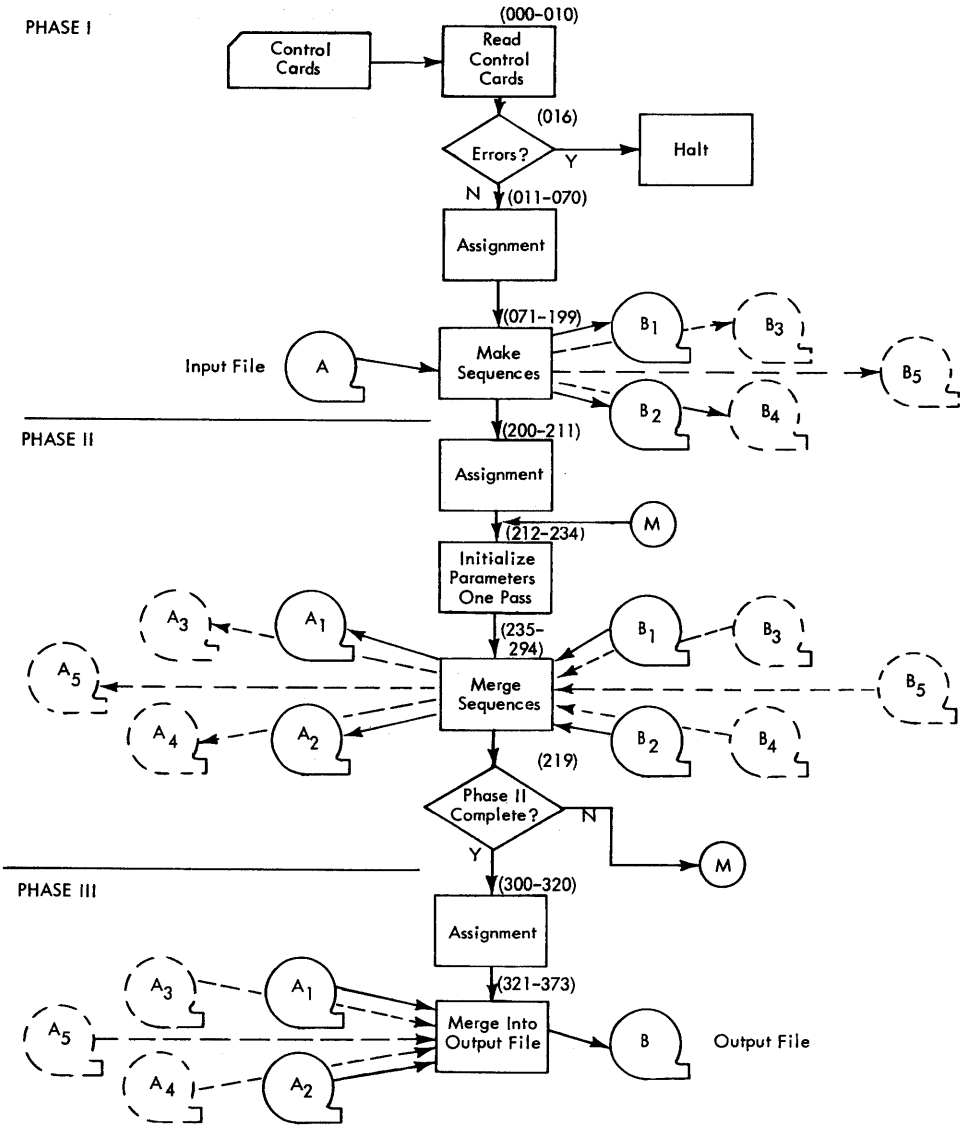


Chart 1A. General Flow Chart

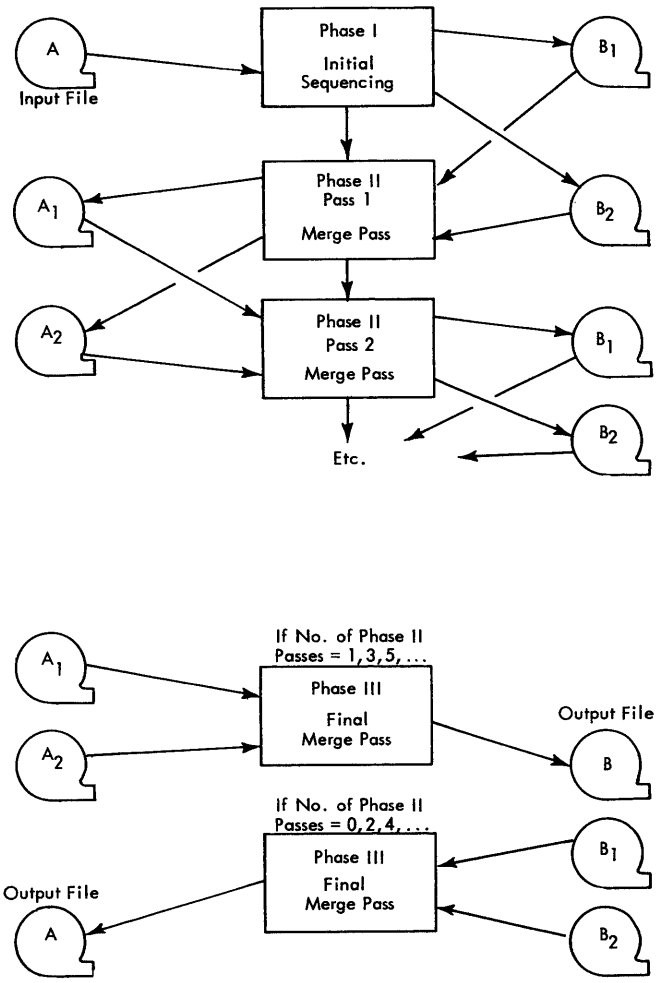


Chart 1B. Input-Output

Phase I

Phase I of Sort 90 has two major functions:

1. The program reads and checks control cards which contain the user's information needed by all phases to adapt the sort into a specific application.
2. Groups of records from the input files to be ordered are read, internally sorted into sequences, and written on work tapes to form the input for Phase II.

Assignment

The program itself is divided into two major parts – an assignment program and a running program. Phase I assignment program performs some services for all three phases and other services uniquely for the running program of Phase I. For the benefit of all phases, it places parameters from the control cards into a communications block, and it computes such vital information as the blocking factor to be used internally by the sort. For Phase I, the assignment program takes information provided by the user together with the constants and skeleton routines it already contains to generate some parts of the running program and to initialize other parts.

Running Program

After assignment is completed, the running program takes over. A number of blocks of records are read into one and then another of two or three large areas of storage. The symbol G is used to distinguish these record areas from the several blocks of records of which they are composed. "G" henceforth refers to any one of these record areas used in Phase I of the sort. Either a two- or a three-area system of overlapping input, output, and processing may be used. With a three-area system, records are sorted in one G, written from another, and read into the third. When all three functions are completed for one set of records in each G, another cycle begins in which the G of the just-sorted records is written, the G just written is filled with new records, and the G just read is sorted. With a two-area system, the program sorts in one G and writes, then reads into the other during a single cycle. These matters are discussed more fully in the section "Input-Output Scheduling." (See Figures 4a and b.)

Sorting consists of several steps. The main parts are a scan and a merge. The scan checks the records for sequences already present in the input file. These "natural" sequences are identified and, if necessary, reversed to conform with the collating sequence specified. Next, a series of passes is made internally to merge these sequences. Each pass performs a series of two-way merges with successive pairs of sequences and reduces the number of sequences by half or more. The merge is concluded when a single sequence remains for the G. The records are then written on an output tape in blocks, which may be different in size from the input or Phase III output blocks. When a sequence break occurs between successive G's, output shifts to another of the M output tapes of Phase I. (M equals the order of merge used in Phase II.) While the first G is being written, the second is being processed. The cycle of events is repeated until the entire input file has been written as a series of sequences which are about equally distributed among the M output tapes of Phase I. These then become the M input tapes of Phase II.

Only RDW's Moved: The key feature of the internal sort, as will become clearer later, is that the records themselves are not moved during the reordering and merging. Instead, only the RDW's (record definition words) of the records are rearranged. This greatly increases the efficiency of the sort, but it also means that after the first read, form 1 and 2 records from a single input block are scattered in storage rather than occupying contiguous locations. Knowledge of this fact is important whenever one examines a print-out of the record area. For form 3 records, a block is read into a contiguous storage area, since new RDW's defining each record are required after each read.

Running Program: Internal Sort

The running program is discussed first, since the function of the assignment program should then be more meaningful to the reader. Whenever the collating sequence is not specified, it is assumed to be ascending.

During the execution of the running program, upper storage contains the following major areas, which begin from the first location after the running program:

1. Three or four areas to contain the lists of RDW's defining the locations of the records – one area for each G, and a spare.
2. A sequence table to hold the sequence-defining RDW's.
3. The two or three G's to hold the records being sorted (Figure 1).

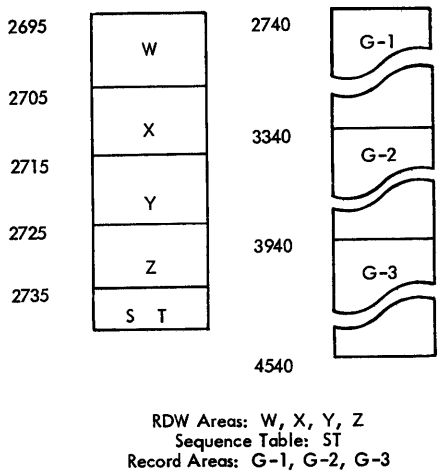


Figure 1. Example, Storage Locations of G's, RDW's, and the Sequence Table

The beginning location varies because the length of the running program depends upon record form and the number of control-data segments. The final location is limited by the amount of user programming in upper storage and the necessity of G's being an equal multiple of the input blocking.

Scan

BASIC PROCESS

The first step in sorting records is the scan. The function of the scan is to discover and define the sequences already existing in the input records occupying a single G, so that these sequences can then be merged. The scan begins by comparing the control data of the first record in the G with that of the second. If a low condition results, the scan records the beginning of an ascending sequence; for a high condition, it records the beginning of a descending sequence. The control data of the second record are compared with those of the third record. If the comparison results as before, the sequence is continuing; if not, a sequence break exists between the second and third records. If the sequence continues, the comparisons between each record and the next continue; when the sequence breaks, the record, the control data of which were responsible for the break in sequence, is considered the first record of a new sequence. If two successive records with identical control data are found, the sequence (ascending or descending) continues. If the first two records of a sequence have identical control data, the sequence is arbitrarily classified as an ascending sequence.

SEQUENCE DEFINERS

The scan defines the sequences it finds by referring to the successive RDW's which locate the sequenced records. It does this by generating a sequence-defining RDW (hereafter referred to as an SRDW) that gives the locations of the first and last record RDW for the given sequence. It enters this SRDW into the sequence table. The first SRDW in this table defines the first sequence of record RDW's, the second SRDW defines the second sequence of RDW's, and similarly down the list. Since any two successive records must form a sequence, the maximum number of entries in the sequence table will equal half or just over half the number of records in a G; e.g., six out of 12 or six out of 11.

INVERTING SEQUENCES

The RDW's of a sequence whose collating sequence is opposite to that specified in the sort are inverted (reversed in order) as soon as this sequence is terminated. The inverted list of RDW's will, therefore, reference records in the specified order. By this technique all sequences in the group (G) to be sorted are in proper collating sequence when the scan is finished. This inversion of sequences is accomplished through the use of the RDW area not currently associated with one of the record areas. When the comparison of the first two records in a new sequence indicates a reversal from the collating sequence, the RDW of the first is placed at the end of the spare RDW area. If the comparison between the second and third records shows that they continue this sequence, the RDW of the second is placed second from the end of the spare RDW area. Each succeeding RDW which continues the sequence is placed in a location one less than the last. When the sequence terminates, the final RDW is placed in the spare area, and the now inverted RDW's are returned to their original location.

Internal Merge

FIRST PASS

When the scan of a G is completed, the "natural" sequences it identified must be merged into a single sequence. This is done through a series of passes of a two-way internal merge. It starts by merging the records of the first and second sequences. This merge begins by comparing the first records of the two sequences and entering the RDW of the lower into the first location of a new list which occupies the spare RDW area. Next, the second record of the sequence whose first was lower is compared with the first of the other sequence and the RDW of the lower enters the new list. With an important modification, discussed below, the process continues until all the RDW's of the first two sequences have entered the new list. The sequences are merged; the new set of RDW's, referenced in their present order, represents the records in

a single sequence. The first and last locations of this set of RDW's are used to generate an SRDW. In the sequence table the first two entries, which defined the just-merged sequences, are no longer necessary, so the new SRDW enters location 1 of this table, overlaying the earlier entry.

The program next merges the records of the third and fourth sequences by adding their RDW's in proper order to the new list. This second set of RDW's represents a second-ordered sequence of records. Its first and last locations are also used to generate an SRDW which enters location 2 of the sequence table, and the third and fourth previous entries are no longer needed. This cycle is repeated for all successive pairs of sequences in the G being processed. If, after merging two sequences, it is discovered that only one sequence remains, the RDW's describing that sequence are simply transmitted intact to the end of the new RDW list and the proper SRDW is generated and entered into the sequence table.

ADDITIONAL PASSES

When one pass of the internal merge is completed, the next begins. This time the original RDW area associated with the G being processed is available for reordering the RDW's, so an alternation between RDW areas from pass to pass is a characteristic of the internal merge. The sequences being merged in any one pass are those which were found as a result of the preceding pass. The number of sequences grows fewer; the length of the sequence table grows shorter. After the last merge pass is completed, a single SRDW remains in the sequence table. The resultant RDW list this SRDW defines may be in either RDW area, depending on the number of passes, and refers to records in a completely sorted G. This RDW list is now ready to control a write operation, and the records in G will be written on tape in a physically sequenced order.

For form 1 and 2 records whose (maximum) length is not changed by editing, the *output* RDW list associated with one G becomes the list controlling input. When editing changes the record length, each RDW in the list is modified to the specified input record length before the read operation. In any case, after the read operation, the records in the corresponding G are not likely to be located in storage in the same physical order as they were on tape. This is not relevant for processing, however, since it is the RDW's which are moved and which will control writing. The first record read is represented by the first RDW in the read list, regardless of which part of the G is addressed. Successive RDW's in the read list represent records in the order read, but the records themselves are scattered throughout the record area (G).

MERGING MORE THAN TWO SEQUENCES

One special feature of the internal merge is a provision for merging more than two sequences into a single sequence in one pass, when circumstances are favorable. At some stage during the merge of two sequences, all the RDW's of one sequence will have entered the new list, but at least one RDW of the other sequence will not have been moved. Each time this happens, the control data of the final record of the sequence which has been completely merged is compared with those of the first record of the next sequence. Should no stepdown (break in the collating sequence) exist between these two records, the program continues by merging the yet unmerged records of the unexhausted sequence and all the records of the next sequence. Since this comparison is repeated when either of these sequences is exhausted, it is possible that many input sequences may merge into any one output sequence during a single pass. If input records are partially ordered, this technique may appreciably decrease the processing time of Phase I.

SEVERAL G'S INTO A SINGLE SEQUENCE

A stepdown check is made before the second and each of the succeeding G's are written. The control data of the last record of the G just written and the first record of the next G are compared. If they indicate a break in the sequence, the output tape is changed. Rotation among the output tapes between sequences tends to equalize the number of sequences on each tape, to the advantage of Phase II. The writing of several G's as a single sequence on one output tape, when there is no sequence break, takes account of partially ordered records and also tends to save time for Phase II.

Example of Scan and Merge

The following example illustrates the operation of the scan and merge of Phase I. In making this example concrete, certain assumptions are made about the amount of storage available for records and its location. It must be remembered that storage allocation will vary with the particular application of Sort 90.

1. Sort into an ascending sequence:
   1000 sixty-word records, fixed length (form 1)
   Five records per block, input
   Two-digit control data
2. Machine components:
   IBM 7070 with 5000 words of core storage
   Two tape channels

The control data are short, and the records are long, so the sort would tend to be tape-limited. Therefore, if the user did not choose to specify it, Sort 90 would have selected a three-area system to allow concurrent reading, writing, and processing. The program computed G's which are a multiple of the input blocking. It is as-

sumed that added instructions for SPOOL or editing are not long and that enough storage is available so that each G will hold two blocks of five records occupying a total of 600 words of storage. A further assumption is that the user specified a three-way merge for Phases II and III and that Phase I assignment has computed an internal blocking factor of three. The user has chosen this order of merge after considering the number of tape units available, the number of passes likely to be saved in Phase II, and the added per-record per-pass processing time required for higher-ordered merges.

Storage beyond the running program contains the records being sorted, the associated RDW lists, and a sequence table. Figure 1 gives memory maps of these as they are assumed for this example. At the start of the first processing cycle, RDW list W is associated with G-1's records. In Figure 2 these RDW's are shown, and each is assigned a number to simplify the following description. In Figure 3A, these RDW's are shown together with the control data of the associated record. Z is the RDW list not currently associated with any G, and therefore available for use as an empty area.

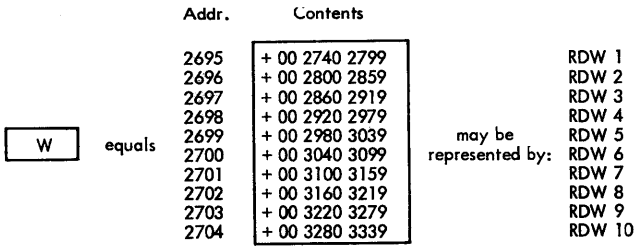


Figure 2. Example, an RDW List

Word in RDW List	A Start In W	A CD*	B End of SCAN In W	B CD*	B Seq.	C I MERGE End of Pass 1 In Z	C CD*	C Seq.	D End of Pass 2 In W	D CD*	D Seq.
1st	RDW 1	03	RDW 2	02	1	RDW 2	02		RDW 2	02	
2nd	RDW 2	02	RDW 1	03		RDW 1	03		RDW 1	03	
3rd	RDW 3	14	RDW 3	14	2	RDW 7	05		RDW 7	05	
4th	RDW 4	17	RDW 4	17		RDW 6	09	1'	RDW 8	06	
5th	RDW 5	15	RDW 7	05		RDW 3	14		RDW 10	07	
6th	RDW 6	09	RDW 6	09	3	RDW 5	15		RDW 6	09	1"
7th	RDW 7	05	RDW 5	15		RDW 4	17		RDW 9	10	
8th	RDW 8	06	RDW 8	06	4	RDW 8	06		RDW 3	14	
9th	RDW 9	10	RDW 9	10		RDW 10	07	2'	RDW 5	15	
10th	RDW 10	07	RDW 10	07	5	RDW 9	10		RDW 4	17	

*Digits of control data in the record defined by the corresponding RDW

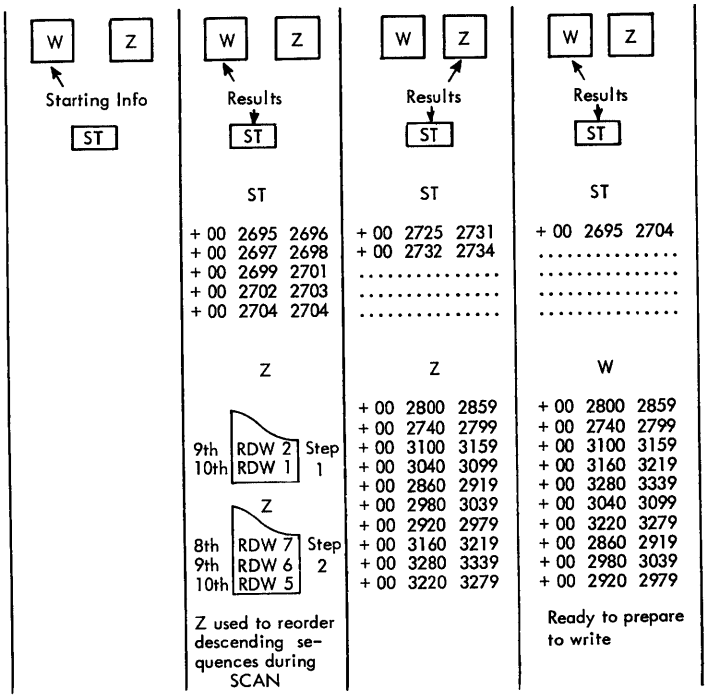


Figure 3

During the scan, five natural sequences are defined, two of them requiring reversal of the RDW's and temporary use of Z. The sequence-defining SRDW's are entered into the sequence table as the sequences are determined. Figure 3B shows the reordered RDW list, the SRDW's in the sequence table, and the successive uses made of Z to invert the RDW's.

In the first pass of the merge, it is possible to merge the five natural sequences into two, since the higher control data of the first record of the third sequence permits it to be merged with the remainder of the second after the first is exhausted. The successive comparisons and sequence definitions made during the first pass of the merge are given in Table I. The results of this pass for the RDW list and the sequence table are given in Figure 3C. The RDW's have been merged into the "empty" area Z.

A second pass of the merge is all that is required to complete Phase I processing for this G. The two output sequences of pass 1 are merged into a single sequence with the reordering of RDW's into area W. Figure 3D indicates the final status of this RDW list and the sequence table. In the next cycle the records from G-1 will be written on the same output tape in blocks of three with the fourth block having only one record.

NOTES

1. The next time records are read into G-1, the RDW's in W will not be in the order of the records in G-1, and this is the typical situation.

2. The RDW area associated with a particular G varies throughout Phase I. In the above examples, if the scan had found only two natural sequences instead of five, the third step would have been unnecessary, and at the end of pass 2, Z would have contained the final ordering of RDW's. This would leave W as the "unused" area when processing the next G.

3. A two-area system in Phase I would require one less RDW area and one less record area (G).

4. After each block is read, the RDW which was minus for purposes of reading that block is made plus. After processing and before writing, the signs on the appropriate RDW's are made minus.

5. The possibility of forming one new sequence from more than two in a single pass frequently allows the number of sequences to be reduced by more than one-half in a single pass and therefore reduces the number of passes for a given G. The example illustrates this for pass 1 of the merge.

Successive Comparisons	Control Data Compared	RDW Associated With	Moved to List Z, Location	
1.	14:2	2	1	Rcd of Seq 1, 2 Compared
2.	14:3	3	2	
3. *	5:3	-	-	Last Rcd of Seq 1 Compared with first Rcd Seq 3.
4.	5:14	5	3	Rcd of Seq 3 and the Remainder of Seq 2 Compared.
5.	9:14	9	4	
6.	15:14	14	5	
7.	15:17	15	6	
8. **	6:15	17	7	Last Rcd of Seq 3 Compared with first Rcd Seq 4.
				sRDW Locating RDW's of first New Seq is Placed in first Word of ST.
9.	7:6	6	8	Rcd of Seq 4, 5 Compared.
10.	7:10	7	9	
		10	10	
				sRDW Locating RDW's of 2nd New Seq is Placed in 2nd Word of ST.

* Sequence 1 is exhausted, but the control data (3) of the last record of Sequence 1 are less than the control data (5) of the first record of Sequence 3. Therefore the new sequence is extended by merging Sequence 3 with the remainder of Sequence 2.

** Sequence 3 is exhausted, and the control data (15) of the last record of Sequence 3 are greater than the control data (6) of the first record of Sequence 4. Therefore the remaining RDW of Sequence 2 is moved into the succeeding word of Z. This terminates the first new sequence.

Table I. Successive Comparisons and Sequence Definitions during Pass I of the Internal Merge

Description of Flow Chart 3: Scan

INITIALIZATION

The scan is entered at AISCAN (071) where the necessary switches are set to N. The index word controlling entry to the sequence table and the one used to build the sequence definers are set to hold the starting locations of the sequence table and the RDW list for this G, respectively.

SEQUENCES IN NORMAL COLLATING SEQUENCE

Index word ARECORD is loaded with the first RDW for this G (072), and the user is given an opportunity to exit at ARECDBRNCH to add routines for editing or deleting records (073). If there is no exit here, the program adds one to the record count (073-1), continues through the N branch of SWA (074), sets SWA to B (075), transfers the RDW in ARECORD to index word ACRX2 (082), and, unless this RDW was the last in the list (090), loads the next RDW in the list into ARECORD (072). Again assuming there is no exit at ARECDBRNCH (073), the program counts the record (073-1) and drops through SWA (074) to the comparison routine (076). Here the control data of the records represented by the RDW's in ARECORD and ACRX2 are compared. If the second record represented by ARECORD is greater than or equal to the first, and if the sort is set for an ascending sequence, the sort continues through the N branch of SWB (077), sets SWC to A (083), transfers the RDW in ARECORD to ACRX2 (082), and again if this RDW was still not last in the list (090), loads the next RDW from the list in ARECORD (072) ready for the comparison (076). As long as successive RDW's represent records with equal or higher control data, this loop through the comparison routine will continue. A low record, indicating a sequence break, will cause the sort to leave the comparison routine (076) through the A branch of SWC, setting and storing in the sequence table an SRDW (088), returning SWC to N (089) and continuing (through 082, 090, 072, 073, 073-1, 074, 076) to load the RDW's and the first two records of the next sequence.

SEQUENCES NOT IN NORMAL COLLATING SEQUENCE

If the first two records form a descending sequence, the sort leaves the comparison routine (076) through the N branch of SWC (084), sets SWB and SWC to B (087), unloads the RDW of the first record into the last position of the spare RDW list (086), and returns (through 082, 090, 072, 073, 073-1, 074, 076) to load the RDW's and compare the second and third records following the termination of the first sequence. If the third record is less than the second record so that it continues the descending sequence, the program proceeds through the B branch of SWC and unloads the RDW of the second record into the second last position of the spare RDW list (085, 086). Blocks (085) and (086) are also reached if the records have identical control data, through the B branch of SWB and the equal branch of the following switch (078). The loop through the comparison (076) continues until the newest record proves larger. The sort then leaves the comparison through the B branch of SWB (077) and the unequal branch of the next switch (078), sets and stores an SRDW in the sequence table (079), returns the inverted list of record RDW's to the proper RDW list (080), resets SWB and SWC to N (081), and transfers the RDW of the first record in the next sequence to ACRX2 (082).

If the user had specified a descending collating sequence for the sort, the high and low branches of the comparison (076) would have been reversed during assignment, equal remaining the same. This is true also of the major comparison in the internal merge, since the same compare routine is used.

CONCLUSION OF SCAN

Whenever the counter (090) indicates that the RDW for the last record has been entered into ACRX2 (082), the program branches to set and store the final sequence-defining SRDW in the sequence table (091), returns the inverted RDW's from the spare area to the proper RDW list—if the last sequence was contrary to the collating order specified (092, 093)— and proceeds to the internal merge.

REENTRIES FROM OPTIONAL EXITS

If the user exits at ARECDBRNCH (073) to do any one-by-one processing, he should reenter at AWRITE (094). This entry leads to the necessary adjustments for any change in record length (095, 096), and the loop continues as otherwise determined from the record count (073-1). If the user deletes the record whose RDW is in ARECORD after an optional exit from ARECDBRNCH (073), however, he should reenter at ADELETE (097) to signal AMOVESW of the merge (103, Chart 4) that a short G is being processed. If this is not the last record of the G (098), this RDW is moved to the end of the list after all below it have been moved up one place (099), and the program loads the next RDW and compares the proper two records (072, 073, 073-1, 074, 076).

When, following a deletion, a check (098) indicates that the last record has been deleted, the program tests to see if any records in G were not deleted (100). If all records were deleted, the sort sets (101) AWDELETESW (166, Chart 5A) in the I-O scheduler to N to bypass writing this G and branches through connector H to the I-O scheduler to prepare for processing the next G. If the last record of the G was deleted (098), but some records in this G were not deleted (100), the program

continues to set and store the final SRDW in the sequence table (091). If the last sequence required inversion (092), it returns the inverted RDW's to the proper list (093) and enters the merge.

Description of Flow Chart 4: Merge

SHORT G

Entering from the scan, the program initializes the counters and switches to be used on the first pass of the merge. The index word counters which control operations on each RDW list and on the sequence table are prepared (102-1), and SWC and SWB are set to N (102-2). AMOVESW is tested (103); if it has been set to Y, indicating a short G, the program branches to reset AMOVESW (104) and duplicates the RDW list in the spare RDW area (105). This is necessary because the merge normally moves from the old list to the new one only the RDW's representing current records. Without the duplication, should the final output list be in the originally "empty" RDW area, the unused RDW's in the input list would be missing. Difficulty would be encountered when this RDW list was next used to read a (larger) group of records into storage. The duplication, which occurs whenever a G is short, insures against such a loss of RDW's. The short G may arise from the reading of one or more short blocks or from the deletion of records during the scan.

LOADING SEQUENCE CONTROLS

After testing AMOVESW (103), and duplicating the RDW list if necessary, the number of sequence definers (SRDW's) is checked (106). If there is more than one entry in the sequence table, the program moves the SRDW for the first sequence into an index word, referred to as "sequence A control" (107). This index word and another, referred to as "sequence B control," count and control the entry of record RDW's into the locations used for comparison and thus into the merged sequence. When sequence A control has been set up (107), it is used to enter the RDW for the first record from sequence A into ACRX2 (108). Then sequence B control is loaded with the SRDW for the second sequence (109) and, under its control, the RDW of the first record in this sequence enters ARECORD (110).

TWO MERGING LOOPS

The control data of the records represented by ARECORD and ACRX2 are compared (112), and (assuming the collating sequence is ascending) the RDW of the lower record is moved to the first location of the empty RDW area. A check is made to see whether this was the last RDW of its sequence, and, if not, the sort returns to load the next RDW into the same comparison word. Depending upon which record is lower, one of two loops is followed to perform these functions:

1. If sequence B record is lower, the sort moves through the N branch of SWC (121), executes the move instruction (122), leaves the sequence B test through branch N (123), loads the next RDW from sequence B into ARECORD (110), and compares this next sequence B record with the sequence A record still referenced by ACRX2 (112).
2. If sequence B record is not lower, the sort moves through the N branch of SWB (113), executes the move instruction (114), leaves the sequence A test through N (115), loads the next RDW from sequence A into ACRX2 (111), and the sequence B record still referenced by ARECORD is compared to this next sequence A record referenced by ACRX2 (112).

On a merging loop, sequence A and its control always refer to the sequence whose SRDW is nearest the beginning of the sequence table. However, when testing whether a third input sequence may join the merge, sequence A and its control refer to the exhausted sequence.

SEQUENCE B EXHAUSTED

One or both of these loops are repeated until the last RDW of one of the sequences is moved to the new list. The program then checks to see if the next sequence can be merged with the remainder of the unexhausted sequence into the current output sequence. Whenever sequence B is exhausted, the program passes through the Y branch of the merge test of this sequence (123) and sets SWC and SWB to T (124). The contents of sequence A control — containing the SRDW for the unmerged portion of sequence A — are transferred to temporary storage (125) and replaced (with interchange) by the contents of sequence B control (126). If more sequences are represented in the sequence table (127), the program branches to load the next RDW for sequence A into ACRX2 (108). In this case the next RDW for sequence A is now the last RDW for the just-exhausted sequence. Sequence B control is set to refer to the next (third) sequence (109), and ARECORD is loaded with the first RDW of this sequence (110). Then the program compares the first record of the third sequence to the last record of the exhausted sequence (112).

SEQUENCE CONTINUES

If the control data of the first record of the third sequence is not lower, the new input sequence can be

merged into the current output sequence. When this happens, the program continues through the T branch of SWB (113), replaces the contents of sequence A control with the sequence definer for the unmerged portion of the first sequence from temporary storage (119), resets SWC and SWB to N (120), loads the next RDW for this sequence into ACRX2 (111), and continues the merge.

SEQUENCE BREAK

If, on the contrary, the control data of the first record of the third sequence are less than those of the last record of the exhausted sequence, there is a sequence break. The remaining records of the incompletely merged sequence must be added to the current output sequence, and the third sequence is readied as one of the inputs for a new output sequence. The program leaves the comparison (112), goes through the T branch of SWC (121), moves remaining RDW's of the incompletely merged sequence to the new RDW list (128), resets SWC and SWB to N (129), and moves the SRDW of the third sequence from sequence B control to sequence A control (130). It enters the SRDW for the output sequence just completed into the next available (first) position in the sequence table, replacing (functionally) those two (or more) SRDW's which defined the input sequences that were merged (130-1). If there are more entries in the sequence table (131), the program branches to load the first RDW of sequence A into ACRX2 (108). It loads sequence B control with the SRDW for the next input sequence (109), loads the first RDW of this sequence into ARECORD (110), and enters the first comparison for merging the next output sequence (112).

SEQUENCE A EXHAUSTED

Whenever sequence A is exhausted, the series of instructions is similar to that indicated above for the exhaustion of sequence B. The program passes through the Y branch of the merge test of this sequence (115), SWC and SWB are set to T (116), and the contents of sequence B control, containing the SRDW for the unmerged portion of sequence B, are transferred to temporary storage (117). If more sequences are represented in the sequence table, sequence B control is set to refer to the next (third) sequence (109) and ARECORD is loaded with the first RDW of this sequence (110). Then the program compares the first record of the third sequence to the last record of the exhausted sequence — still referenced by the contents of sequence A control (112). If the sequence continues, the program proceeds as previously described through the T branch of SWB, AMCONTINUE, etc. If there is a sequence break, the program proceeds as previously described through the T branch of SWC, AMBREAK, etc.

NO MORE INPUT SEQUENCES

1. If there are no more entries in the sequence table after a sequence break (131), the program moves the SRDW of the new input sequence from sequence A control to temporary storage (132) and branches to the series of instructions, beginning with AMFINIS (133) which concludes a pass of the merge.
2. If there are no more entries in the sequence table after sequence B is completely merged (127) or after sequence A is completely merged (118), the temporary storage area has already been loaded with the SRDW for the unmerged portion of the last input sequence (125 or 117); the program branches directly to AMFINIS (133) in either of these cases. At AMFINIS the remaining RDW's of the last sequence are moved to the new RDW list (133); the last new SRDW is entered into the sequence table, and the old RDW list is set to become the "empty" area for the new RDW list of the next pass (134).

STEPDOWN CHECK

At the beginning of the major loop (102-1), switches and counters are reinitialized in preparation for another pass. The program drops through the N branch of AMOVESW (103) and a decision is made as to whether another pass is necessary or not (106). If another pass is necessary, the program continues as before. If, however, the sequence table contains only one entry, thus indicating that all the records in the G are in sequence, the sort branches to a stepdown check (135).

If the control data of the first record in this sequence are equal to or greater than those of the last record in the previous sequences, this G will be written on the same output tape as the earlier one. But if the control data of the present G are less, they will be written on a different output tape (136). If a two-area input-output system is used, a separate area is reserved to hold the last record of each sequenced G, so that it may be saved for comparison with the first record of the next G after the latter is sequenced. This is not necessary in a three-area system, since the area being written while the current G is processed is not destroyed until the latter processing is complete.

OUTPUT PREPARATION

The last step in the internal merge is the preparation of the sequenced records for transfer to tape. This includes changing the signs of the RDW's in accordance with the internal blocking (137). Then, entry is made to the input-output scheduler.

Input-Output

Scheduling

Although IOCS routines incorporated in Sort 90 provide many standard I-O routines and tape scheduling, in Phase I the sort program itself adds two important provisions:

1. It programs for the options which IOCS leaves to the user, such as the treatment of unusual conditions (SCLR, LLR, and EOS) and additions to end-of-reel routine. (The user may still add his own coding to the end-of-reel routine at appropriate points.)
2. Even more important (in Phase I only), the sort must provide the coordination to insure that all necessary activity of one cycle is completed before another is initiated. In one normal cycle all records from one G must be internally sorted, all records in another G must be written on tape, and this second G, or another, must be filled with unsorted records. The proper sequence of events must also be insured when the sort first begins and there are no records to be written or processed during the first cycle. Similarly, input end-of-reel (EOR) procedures must be deferred until all records from the current input tape have been sorted and written.

SCHEDULING FOR PHASE I PROCESSING

Because of the sorting method used in Phase I, the GET and PUT macro-instructions are not used. In normal input-output scheduling, each GET prepares one record for processing and signals the file scheduler whenever a block is empty. The PUT prepares one record for output and signals the file scheduler whenever an output block is ready to be written. With such usage, processing must be done on a record-by-record basis. In Phase I of Sort 90, processing is done on a record group basis. Therefore, it has been necessary to substitute a specialized set of instructions to ready groups of records for processing and writing and to signal the file schedulers. However, the power of the sort in group processing sacrifices flexibility in some respects. In the serial processing required with GET's and PUT's, it is relatively easy to insert or delete records. Not so with group processing. In practice, insertion is impossible, and deletion requires the special technique of reproducing the RDW list to prevent loss of RDW's. See ADELETE (097) on Chart 3 and AMOVESW (103) on Chart 4.

PRIORITY OPERATIONS WITH G'S

The problem of moving records in and out of storage during Phase I of Sort 90 is somewhat unusual because the current input, output, and work areas commonly consist of many blocks of records rather than one. These large G's are desirable because they make possible larger sequences for the output of Phase I. Therefore, input-output programming of the sort is coupled with standard IOCS routines in such a way that any number of blocks are moved in and out of storage efficiently during a single processing I-O cycle. This multiple-block processing means that as soon as one block of records is either read or written, priority routines must be available to detect and process errors, to modify counters and switches, and to initiate the next read or write instruction as long as any records of a single G remain. The 7070's priority-processing features make it possible to interrupt the sorting process temporarily while such priority routines are executed.

TWO OR THREE-AREA I-O

Sort 90 gives the option of either a two or three-area system of overlapping input, output, and processing. In a two-area system, tape operations are overlapped with processing but reading is not overlapped with writing, while a three-area system permits all three to occur at one time – although in any case they must be initiated serially. In a two-area system, one G of records is sorted while, in another G, already-sorted records are written out and, following this, unsorted records are read during the same cycle. In a three-area system, sorted records are written from one G, unsorted records are read into another, and the sorting occurs in the third, during a single cycle. Since the sorting occurs through the rearranging of RDW's, the records themselves are not moved about in storage. Figure 4a and b illustrates the two and three-area systems.

If the sort tends to be tape-limited (i.e., it takes longer to read and write records than to sort them), the three-area system is desirable. If, on the other hand, many segments of control data and short records combine to lengthen process time beyond the combined read and write times (a process-limited application), then a two-area system may be preferable. The user may specify which system the sort is to use, or he may leave it to the assignment program to decide which is better.

NORMAL CYCLE

Sort scheduling must take into account three principal types of conditions and provide the proper sequences of instructions peculiar to each. First, there is the normal sequence of events when sorting is underway. The just-sorted records must be written, unsorted records must be read, and other records must be sorted during a single cycle. (For example, see cycles 3 and 4

	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7
G1	Read	Proc.	Write	Read	-	-	-
G2		Read	Proc.	Write	-	-	-
G3			Read	Proc.	-	-	-
	a. Three-Area System						
G1	Read	Proc.	Wr-Rd	Proc.	-	-	-
G2		Read	Proc.	Wr-Rd	-	-	-
	b. Two-Area System						
G1	Write	Read	* Proc.	Write { EOR Routine }		Read	Proc.
G2	Proc.	Write	Read				Read
G3	Read	Proc.	Write		Read	Proc.	Write
	c. Three-Area System, EOF Condition, EOF encountered in Cycle 3 before any records read.						
G1	Read	* Proc.	Write	Write { EOR Routine }	Read	Proc.	Write
G2	Write	Read	Proc.			Read	Proc.
G3	Proc.	Write					Read
	d. Three-Area System, EOF Condition, EOF encountered in Cycle 2 after some records read.						
G1	Wr-Rd	Proc.	* Wr-Rd	Write { EOR Routine }	Read	Read	Proc.
G2	Proc.	Wr-Rd	Proc.			Proc.	Wr-Rd
	e. Two-Area System, EOF Condition, EOF encountered in Cycle 3 before any records read.						
G1	Proc.	* Wr-Rd	Proc.	Write { EOR Routine }	Read	Proc.	Wr-Rd
G2	Wr-Rd	Proc.	Write			Read	Proc.
	f. Two-Area System, EOF Condition, EOF encountered in Cycle 2 after some records read.						

* Indicates cycle in which EOF signal is encountered.

Figure 4

of Figure 4a and b.) Write instructions and counters are set up for the G to be written, and the write instruction for the first block is issued.

If a *three-area* system is used, read instructions are set up for another G, and the read command for its first block is given. Then the program begins to sort the records in the remaining G through the scan and internal merge. Whenever a tape operation is completed, priority is signaled. Sorting is temporarily interrupted, and a priority routine is entered to correct a tape error (if any), modify the appropriate switches, and issue another tape read or write instruction. Then sorting resumes, unless another signal for priority has been given. Counters in the priority routines cause the sort to bypass the tape command and give a priority release whenever the tape operations on a G have been completed. This terminates reading and writing at the proper time. Reentry to I-O programming from the scan and merge is delayed by gates which are not opened until tape operations are complete. Sorting and tape operations must each wait for the other to conclude before the next cycle is begun.

Procedures are similar for a *two-area* system except that the program enters processing directly after the first write instruction is issued, and the first read instruction is given only after the tape-write priority routine indicates that all the records in the G have been written.

BEGINNING-OF-SORT CYCLES

A second situation is encountered when the sort begins and there are no records to be sorted or written. See cycles 1 and 2 of Figure 4a and b. This condition occurs just after the assignment program is completed and processing starts, and also whenever a new input reel is first processed. Reading begins at once, and switches are set to prevent any activities except priority read routines until the first G is filled. A full G terminates the first cycle. The second cycle bypasses the write instruction, sets the read instruction for a new G, issues the instruction to read the next block into this G, and starts sorting the records in the first G. Sorting is temporarily interrupted by the read priority

routine each time a block has been read. Switch settings have been changed so that when all records in the first G have been sorted, and the second G is full, the program is ready for the normal condition at the beginning of the third cycle.

INPUT END-OF-REEL CYCLES

The third situation confronted by the scheduling routines is the handling of the input end-of-reel condition. To allow the checking of record counts, all unsorted records in storage must be sorted and written before the end-of-reel routine is entered. If end-of-reel is discovered by the priority routine before any records are read during a given cycle, switches are set to limit the next cycle to writing the records being sorted in this cycle. When these records have been written, the end-of-reel routine is entered. If end of reel is encountered after some blocks have been read, switches are set to limit the sort to process and write during the next cycle and to write only in the cycle following it. The end-of-reel routine is entered in the third succeeding cycle. (See Figure 4c, d, e, f.)

Description of Flow Chart 5A

NORMAL CYCLE

During a normal cycle the entry is from the scan and merge to a switch which is set to loop on itself until the last block of the G being written has been written (160). When writing is complete, the write block counter is reset (161). If the last sequence processed breaks with the previous sequence (161-1) and if M−1 output reels contain only a single sequence each (161-2), then HALT 1114 follows (161-3). If alteration switch 1 is set off, the sort will drop the last input reel (168-4) and proceed to terminate Phase I. Since the number of sequences equals the order of merge, the processing of Phase II will be bypassed. Despite the large number of records, there is no risk of a non-ending sort, and the sort will be completed successfully. If, on the other hand, alteration switch 1 is on (161-4) and sorting continues in Phase I, a non-ending sort will result in Phase II unless records are eliminated by summarizing. If alteration switch 1 is on or if there were not M−1 full output reels containing single sequences, and if the last tape write operation did not reveal an EOF (161-5), then the output is switched to the next active tape (161-6); the just-processed sequence will be written on a different output reel. If the last write operation did find an EOF, then the output end-of-reel routine has already switched the tapes to be written, so this step (161-6) is bypassed, as it is if no stepdown was found between the sequence just processed and the previous sequence (161-1).

In all cases that continue the internal sort, the next step is to modify the write instructions to refer to the G just sorted and next to be written (161-7). If records are still being read on the last cycle, the program loops on a switch until reading is complete (162). AEORSW (163) has been set to N, so the program checks ATURNONSW (163-1) for SLR on previous read and sets AMOVESW (163-2) if it is SLR. The controlling index words are reloaded to rotate the read, write, and process areas for the present cycle (165). For example, in a three-area system, the index-word contents are exchanged so that the G just read will be processed, the G processed written, and the G written read. Normally AWDELETE (166) is set to Y, so the next instruction is a write command (171), which turns over the actual monitoring of the tape operation to the proper channel control. Since the program is not in priority mode, the priority release (172) is executed as a NOP. AGOEORSW is off, so the sort continues through the Y branch of this switch (173), and branches at AAREASW1 according to whether the I-O system has two or three areas (175). If it has two areas, the program generates RDW's if necessary (157-1) and exits to begin the scan; if it has three areas, the program branches to ARSWQ (138) to prepare for reading.

With a three-area system, the program continues through the N branch of ARSWQ (138). If editing which changes the record length is done in Phase I, the RDW's that are modified in editing must be reset to handle the unedited input records (139). The read instructions are loaded from index words to specify the proper G to be read and are otherwise prepared (140). ARTAPEMARK (154) is set (141) to give special handling if an EOF signal is given before any records are read into the current read G, and ARDSW (158 and 162) is set (141) to prevent a reentry from the scan and merge should processing be completed before reading. The first read instruction is given (150), and since priority release (151) acts as a NOP, the program branches through the Y branch of ARXSW (157) and exits to the scan and merge, but only after generating record-defining RDW's (157-2) in the case of form 3 records (157-1).

WRITE PRIORITY ROUTINE

Whenever priority is signaled on the write channel, processing is interrupted at the completion of the current instruction, the contents of the instruction counter are stored in index word 97, and the program branches in priority mode to the write priority routine (167). The first task is to analyze the final status word whose

address is in location 99 to discover the results of the write operation. If condition code 2, indicating a correct length record (CLR), is present in position 1 of the final status word, the program proceeds normally. Other condition codes (except 5) are referred to IOCS to be handled as it has been directed. Condition code 5 (EOF) leads to an output end-of-reel routine, which is described in the following section. After the end-of-reel routine (unless Phase I is terminated) or under normal condition (CLR), the program prepares the instructions to write the next block (169). It then branches to write the next block (171) unless the last block in G has been written (170). A priority release instruction is given (172). This returns the program to non-priority and the instruction whose address is in index word 97, unless the read priority routine has been signaled.

Whenever, in the write priority routine, the test of the block counter (170) indicates the last block has been written, the sort branches to set AWRSW (160) to Y (177), which opens one of the gates to permit entry from the scan and merge when non-priority processing reaches this point. Normally, AGOEORSW is off (178), so the branch is through Y to AAREASW2 (179). With a three-area system, priority is released, and the program returns to the point of interruption or another priority routine. With a two-area system, however, the completion of writing indicates that reading may begin. So, still in the priority mode, the program branches to ARSWQ (138), and the routine to initiate the first read is carried through as described earlier for a three-area system (138, 139, 140, 141, 150). This time, however, the program is in priority mode, and the priority release (151) following the tape read command returns control to the point of interruption in non-priority processing.

OUTPUT END-OF-REEL ROUTINE

When, on a write priority interrupt, an EOF is indicated, the sort branches to an output end-of-reel routine. The proper location in the tape table is tagged (168) to indicate that the reel is temporarily end-of-file but may be backspaced if the current input reel is restarted or dropped from the sort. (See "Input End-of-Reel Routine.") If there are fewer than M−1 full output reels (168-1), writing is switched to the next active output tape (169-1). If, however, there are M−1 full reels (168-1, 168-2) and only M−1 sequences (168-5), the sort sets (168-8) one switch to allow HALT 1114, sets another to allow HALT 1112, and sets (168-9) a third to allow HALT 1107. It then changes output to the last active output tape (169-1).

When there are M−1 full reels (168-1, 168-2) and more than M−1 sequences (168-5), the sort comes to HALT 1110 (168-6). The user should set alteration switch 1 off (168-7) and drop the current input reel (168-4), because he runs the risk of a non-ending sort in Phase II. However, anticipating extensive summarizing in Phase II, he may continue by setting alteration switch 1 on. Switches are set (168-9) to allow HALTS 1112 and 1107, and the sort is readied to write on the last active output tape (169-1).

When there are M full output reels (168-2), the sort comes to HALT 1112 (168-3), which signals the user that the last input reel must be dropped before the sort can proceed to Phase II. The current input reel is dropped (168-4) by backspacing to a segment mark all output tapes except those which became EOF before reading began on the current input reel. A message indicates the drop, and the sort branches to the end-of-phase routine (197).

READ PRIORITY ROUTINE

With a read priority interrupt, the sort enters in priority mode to do a tape condition analysis (142). ATURNONSW (163-1) is set to Y if SLR is indicated. This causes the AMOVESW (103) of the merge to be set later (163-2) when the G being read is processed. When the G remains unfilled (148), ARTAPEMARK (154) is set to N (149), and instructions are modified so that the next block from tape will be read into the next available G locations. The routine returns to the channel scheduler where it gives the next read instruction (150) and a priority release (151). Whenever G is found to be filled (148), ARDSW (162) is set (156) to Y, which drops the second barrier to the start of the next cycle; reading is complete for this cycle, and a priority release is given (151).

If on a read priority interrupt a tape condition analysis (142) reveals an error (condition code 1), a branch is made to IOCS for processing that the user has specified on the first control card (143). If a long length record (LLR), a short character length record (SCLR), or a segment mark (EOS) is revealed (143), the program comes to HALT 1124, 1127, or 1126, respectively (144). When START is pressed, the sort returns to issue the next read command (150) and priority release (151) but, except for the segment mark condition (145), first types out the block involved (146) and drops it from the sort.

INPUT END-OF-REEL CYCLES

Input end-of-reel cycles begin from the interrupt for read priority. (See Figure 4c, d, e, f.) If the tape condition analysis (142) indicates a tape mark (EOF),

the program branches to ARTAPEMARK (154). If the interrupt follows the first instruction for the current G, AEORSW (163) is set to Y (153), an action which in the following cycle causes another switch to be set which will prevent entry to the scan and merge and will lead to the end-of-reel routine at the proper time. This branch is made because no records have been read this cycle and only writing will be necessary in the next cycle before the end-of-reel routine is entered. If, however, end-of-file recognition follows later read instructions, ARTAPEMARK (154) directs the sort through N branch where ARSWQ (138) is set to Y (155) so that AEORSW (153) will be set to Y in the next cycle rather than in this one; records have been read in this cycle which need to be processed in the next. Since reading is complete after either of these branches from an EOF condition, they both return to set (156) ARDSW (162) to Y, dropping the second barrier to the start of the next cycle. A priority release is given (151), and, if any processing or writing remains, this is completed during the remainder of this cycle.

The following cycle (after an EOF is encountered) varies according to whether records were read during the earlier cycle or not. If records were read, writing and processing are to occur this cycle and only reading is to be bypassed. AEORSW (163) is still set to N, so the instructions following the entry from the scan and merge are executed as in a normal cycle – until ARSWQ (138) is encountered after the first write command. This switch (138) has been set to Y, so it is reset (152); AEORSW (163) is set (153) to Y; ARDSW (162) is set (156) to Y; and the priority release (151) functions as a NOP. (The AEORSW setting will eliminate processing from the following cycle, and the ARDSW setting opens one of the gates controlling entry to the next cycle.) Then, through the Y branch of ARXSW (157), the program enters the scan. When all writing and processing are complete, the sort is ready for the third cycle.

During the third cycle, only writing is permitted. Entry from the scan and merge (160) proceeds normally except that AEORSW (163) is at Y setting, so AGOEORSW is set to N, and AEORSW is reset to N. This means that, after the first write command (171) and the priority release (172) which acts as a NOP, the program leaves the N branch of the next switch and executes a one-instruction loop (174). This loop is temporarily interrupted for write priority routines (167) until all writing is complete. In the priority routine which follows the writing of the last block in the G, the program branches (170) to set AWRSW to Y (177) to open one of the gates at the entry from the scan and merge. Since AGOEORSW is on, it is turned off and the N branch is taken (178). The program releases priority (181) directly to the input end-of-reel routine (182).

BEGINNING-OF-SORT CYCLES

When the assignment program is completed, the running program is entered through the scheduling routine. This first entry is made at ARSWQ (138). The program proceeds as in a normal cycle after it has reached ARSWQ. RDW's are adjusted if editing changes record length (139); read instructions are set up (140); and several switches are set (141) before the first read command is issued in non-priority mode (150). The priority release acts as a NOP (151), but now ARXSW is at N setting (in contrast to the normal cycle) and the branch is to a switch (158), which has just been set to loop on itself. Since no G contains records ready for processing or writing, there is nothing for the sort to do but wait until a G is filled with records. Temporary interruptions come from the read priority routine (142). When one of these routines discovers that G is full (148), ARDSW is turned off (156), and after the priority release (151) the non-priority program leaves its one-instruction loop (158) and resets ARXSW (157) to Y. The program is ready to begin the second cycle.

The second cycle is to read and process, but not write. It begins at AEORSW (163), which is set to N. The index words defining the next read, write, and process areas are loaded (165) as in a normal cycle, but AWDELETE (166) has been set to N (during assignment or the end-of-reel routine) to allow writing to be bypassed. AWDELETE is reset (176), and AWRSW is set to Y (177) to open one gate at the entry from the scan and merge. The program drops through the Y branch of the next switch and returns (179, 180) to ARSWQ (138). The first read is initiated from there as in a normal cycle (138, 139, 140, 141, 150). Any priority releases encountered (180, 151) act as NOP's, since the program is in non-priority mode. In this second cycle, the sort leaves ARXSW (157) through the Y branch, generates RDW's (157-2) for form 3 records (157-1) and exits to the scan and merge. When reading and processing are complete, the next cycle, a normal cycle, begins (160).

NOTES

1. Beginning-of-sort cycles occur after input end-of-reel for all except the last input reel. An end-of-reel routine which recognizes the last input reel (by means of the label or a reel count) causes a branch to an end-of-phase routine, which is completed by a call to load Phase II.

2. Whenever user coding added to the scan causes all the records in a given G to be deleted, there are no records to be written during the following cycle. Under these circumstances the scan sets AWDELETE (166) to N, and in an otherwise normal cycle writing is by-

passed in the same manner as in the second cycle of the beginning-of-sort cycles discussed above.

End-of-Reel and End-of-File

The input end-of-reel routine has the following major functions:

1. When labels are used, it checks for discrepancies between the information provided in the input trailer label and the actual number of blocks and records processed; it also checks hash totals if these are used. If a discrepancy is discovered, it gives the user the opportunity to reprocess (restart) the current input reel, to drop it from the sort, or to ignore the discrepancy.
2. The end-of-reel routine rewinds the current input tape and sets up for the next input reel. To do the latter it changes the input file unit number in the DTF and carries out the header label processing for the next reel.
3. The end-of-reel routine provides optional exits where the user may branch to modify IOCS input file treatment or to process additional trailer labels.
4. The end-of-reel routine tests for the last input reel, and, when found, it causes a branch to the end-of-file routine.

REPROCESSING OR DROPPING INPUT REELS

If an input reel is reprocessed or dropped from the sort, it is necessary to "remove" the records of this input reel from the output reels on which they were written. So that only these records will be eliminated, the end-of-reel routine writes segment marks on each output reel between the records from different input reels. It writes these segment marks after it has been determined that an input reel is accepted by the sort. A segment mark is also written on each output tape during assignment before any records are written. When in a later end-of-reel routine an input reel is to be reprocessed or dropped from the sort, the active output reels are backspaced to the previous segment mark. To determine which output reels are active and which were filled before the current input reel was processed, it is necessary to allow output reels to occupy three different statuses: EOF while a previous input reel was being processed (inactive), EOF while the current input reel was being processed (temporarily inactive), and not EOF (active). When an output reel becomes full, it is made temporarily inactive. At input end-of-reel time, it is made permanently inactive, if the input reel is accepted for the sort, or restored to active status, if the current reel is to be dropped or reprocessed.

INCONSISTENCY IN END-OF-FILE INDICATORS

The sort provides two ways of determining end-of-file, if labels are used. The user may punch the number of input reels on his control card, and he may depend upon trailer information. The two indicators may not agree as, for example, if the user punches the wrong number of reels or, in running, omits a reel (intentionally or not). When only one shows end-of-file, the end-of-reel routine provides halts which allow the user his choice of action at that time.

END-OF-FILE ROUTINE

The end-of-file routine concludes Phase I and provides the linkage to Phase II. It rewinds the last input tape, closes the output tapes, types the record counts, and calls Phase II.

Description of Flow Chart 5B

The input end-of-reel and end-of-file routines are described in Chart 5B. The Phase I input end-of-reel routine uses the IOCS end-of-reel routine and the exits it provides. The blocks on Chart 5B which are part of the IOCS end-of-reel routine are indicated by one asterisk (*).

END-OF-REEL WITH LABELS: CHECKING COUNTS

If the user specifies that the input file has labels (182), the end-of-reel routine begins (in IOCS) by comparing the block count indicated by the trailer label of this input reel with the count kept by the sort (183). Next, if the initial comparison showed that the number of blocks processed did not coincide with the number shown on the trailer label (184), the sort leaves the end-of-reel routine through IOCSEX6 and sets (184-1) ARESTARTSW (186) to Y. The sort also checks for discrepancies between the label indication and the hash and record counts made while processing this reel (185-1), if record and/or hash counts were specified (185). When discrepancies are found, a message to this effect is typed and ARESTARTSW is set to Y (185-2).

END-OF-REEL WITH LABELS: NO DISCREPANCY

If the count checks show no discrepancies, the sort drops through the N branch of ARESTARTSW (186), adds the counts to storage for use by Phase II, and writes segment marks on all active output reels (187); this separates records that came from the current input reel from those of following input reels, and it aids in dropping or reprocessing later input reels if discrepancies should be found in them. Some output reels may have become full during the processing of this input reel. Since no discrepancies were found, these reels may be

safely changed from a temporarily inactive status, which was assigned during the output end-of-reel routine, to a more permanently inactive status for the rest of Phase I (187-1). These reels are set so that they cannot be backspaced for the restart option should this be chosen when another input tape becomes end-of-reel. Backspacing under segment mark control of all output tapes not permanently inactive is also necessary for the option of dropping the records of the current input reel after HALTS 1110, 1112, or 1114. The sort gives the user the option of exiting at AMORETRLR (188) for added processing of trailer labels and sets (188-1) SWRB (194) to N, which indicates to a later stage that the present input reel is not to be processed.

The sort may test for the last reel of the input file in two ways. It may check the trailer label; or, if the user has called for a reel count, it may check the reel counter. With these two tests there are four ways the sort may reach the end-of-file routine when labels are used:

1. If there is no reel count, and the trailer label indicates end-of-file, the program branches Y (189) to the end-of-file routine (197).
2. If there is a reel count, if the reel counter is zero, and if an earlier end-of-reel routine has come to HALT 1104, transfer to the end-of-file routine (197) occurs (189). The count of zero indicates that the number of reels specified by the user has now been processed.
3. If both tests (of trailer label and of reel counter) show that the present input reel is the last, the sort branches (189-1) to the end-of-file routine (197).
4. If both tests are made, but one and only one shows an end-of-file condition, the sort types a message indicating this and comes to HALT 1104 (189-2). If the user places alteration switch 1 off, the sort branches (189-3) to the end-of-file routine (197).

If alteration switch 1 is turned on (189-3) after HALT 1104, or if neither indicator shows an end-of-file condition (189-1), the just-processed input reel is not the last, and the sort encounters a switch (189-4) which determines whether HALT 1107 will occur (189-5). If either HALT 1114 (161-3) or HALT 1110 (168-6) has occurred, all input reels but one are full. The user has continued after these halts, and now that another input reel is completed, he is again given the option to terminate Phase I. If, after HALT 1107, alteration switch 1 (189-6) is off, the sort branches to the end-of-phase routine (197). But if alteration switch 1 is on, the sort re-enters the IOCS end-of-reel routine, to which it branches directly (189-4) if neither HALT 1114 nor 1110 has occurred. The block counter is reset to zero for the next reel, the input tape is rewound, and the unit number for the active input tape is changed in the DTF to the number of the unit specified as alternate (190). SWRA was initialized to N (191), so the sort leaves IOCSEOREX for further preparation for new input. Since there are as many as five input tape units and IOCS provides for only two alternates in the DTF, the sort replaces the just-used unit with the next input unit in the tape table (192). The tape table is a list by channel and unit of the tapes which are used by the sort. For the modification just mentioned, the sort enters the number of the next unit in the DTF locations for ALT1TAPE and ALT2TAPE. The reel-by-reel hash and record count fields (if any) are set to zero (193), and, since SWRB (194) was set to N, the user is given the opportunity to exit at AENDOFREEL for added coding (194-1). The sort returns to the IOCS end-of-reel routine to process the header labels of the next input reel (194-3, 194-4) and leaves this routine to enter the I-O scheduler for the beginning-of-sort cycle.

END-OF-REEL WITH LABELS: DISCREPANCY

If ARESTARTSW (186) was set to Y, indicating a count discrepancy, the program resets ARESTARTSW to N and comes to HALT 1101. If alteration switch 1 is turned on, the sort ignores the discrepancy (186-2). If, however, alteration switch 1 is off, any output reels which were put on temporarily inactive status by becoming EOF while the present input reel was being processed are reactivated (186-3). The records which they received from the last input reel will be ignored and also overlaid, if more records are processed. HALT 1102 follows. The user may elect to drop this input reel and its records from the sort by setting alteration switch 1 off and pressing START (186-5). All output reels which are active (not permanently inactive because of EOF) are backspaced to the previous segment mark (187-2). This segment mark was written by the EOR routine when previous input reels were processed and their records accepted for the sort (187). Under the option of dropping the current input reel, the program continues from AMORETRLR (188) to the end (194-4) as was described for the case of no discrepancies. This includes changing the DTF at ARCONTINUE (192).

If the user wishes to reprocess this input reel rather than drop it, he turns alteration switch 1 on (186-5). As with the other alternative, the sort backspaces the active output reels to the previous segment mark (186-7). It rewinds the input reel (186-6), restores the previous reel sequence number (186-8), sets SWRA and SWRB to T (186-9), and zeroes the reel-by-reel block counter. It reenters the IOCS end-of-reel routine but immediately exits through IOCSEOREX to zero the temporary record and hash counters (if any) (193), since

SWRA (191) was just set to T. It bypasses the AENDOFREEL exit because SWRB (194) was also just set to T. It resets SWRA to N (194-2) and returns to the IOCS end-of-reel routine to reprocess the header label on this input (194-3, 194-4). The end-of-reel routine concludes with a branch to the beginning-of-sort cycle.

END-OF-REEL WITH NO LABELS

If the user has specified no labels, the sort exits through IOCSEOFEX, and adds the contents of the reel-by-reel hash and record counters (if any) to the accumulating totals (195). Segment marks are written on all active output tapes (195), and reels which became EOF during the current input reel are set so that they cannot be backspaced (195-1). No restart is possible with unlabeled tapes in Phase I, but these actions permit the option of dropping the records from the current input reel after HALTS 1110, 1112, or 1114. The only test for end-of-file is the reel count, which the user must have specified in the first control card. If the counter equals zero, indicating that the last input reel has been processed (196), the sort branches to the end-of-file routine (197). Otherwise, it branches to a switch (189-4) which will allow HALT 1107 (189-5) under the same conditions as indicated above for labeled tapes with no count discrepancies. Unless the user chooses to terminate Phase I after HALT 1107, the sort returns to the IOCS end-of-reel routine to zero the reel block count and rewind the input tape (190). It also changes the active tape unit address in the DTF to the next tape, which is now specified in ALT1TAPE (190). Because SWRA is initialized to N, the sort exits from the IOCS end-of-reel routine (191) to enter another tape unit number in ALT1TAPE and ALT2TAPE locations (192). It zeroes temporary hash and record counts (if any) (193) and gives the user an optional exit at AENDOFREEL (194-1), since SWRB (194) was initialized to N. There are no header labels to process (194-3); therefore, the sort branches to the beginning-of-sort cycle.

END-OF-FILE ROUTINE

The end-of-file routine begins by rewinding the input reel just processed (197). It gives the user an opportunity to exit at AWINDUP (198) for the final processing and closing of added tapes or for other added routines. Output tapes are closed, and messages are typed which state the number of records successfully read by Phase I, the number of records deleted (if any), the number of blocks dumped for error (if any), and the number of records written (if any were deleted) (198-1). The user is given another optional exit at AENDPHASE1 (199). The sort calls for Phase II (199-1), and, after it is loaded, enters the Phase II assignment program.

Assignment Program

General Description

Sort 90 begins with the assignment program of Phase I. This assignment program must read the user's specifications for the sort application; some of this information it will move directly to a communications block where it is preserved for use by the assignment program of each phase. Other information will be used for computations the results of which are stored in the communications block. Besides these tasks which serve all phases, the assignment program of Phase I also must prepare the Phase I running program according to the user's specifications.

Phase I assignment is sufficiently long that it must be carried out in two parts. In loading the first part, an area immediately following the load program is reserved for the communications block, the skeleton of the running program is placed in its proper position, and two large blocks of the assignment program are entered, one located before the running program and the other after it. The first part reads and checks the control cards, enters the user's specifications into the communications block, generates the routines used to compare the control data of the records to be sorted, calculates the storage available for records, and, if desired, decides whether a two- or three-area system is to be used for Phase I. For the second part, IOCS routines and another section of assignment are loaded. The assignment and IOCS OPEN routines are entered over the second section of part 1 of assignment, where they in turn will be overlaid by the reading of records during the running program. The remainder of IOCS is entered over the first section of part 1 of assignment, where it is retained through all three phases of Sort 90. This second part of assignment opens the input and output tapes, zeroes upper storage, and generates the RDW's to define the records to be sorted. (See "Memory Maps," Figure 5.)

Description of Flow Charts 2A and 2B

CHECK OF PHASEKEY

When part 1 is loaded, the first step is a test (000) of the Phasekey to determine whether this is a restart. If the Phase II constant is in APHASEKEY, this sort is probably being restarted after an interrupted Phase II. If this is a restart, the program comes to HALT 1010 (001). If alteration switch 1 is on (002), or if APHASEKEY did not contain the Phase II constant, the sort proceeds to the assignment program and issues the message "7070 SORT 90" (003).

When alteration switch 1 is off (002), the program comes to HALT 1011 (002-1). If alteration switch 1 re-

mains off (002-2), the program branches to load part 2 of Phase I (053). However, the user may be loading his sort program from cards; if so, it will be necessary to go through the control cards before loading part 2. Setting alteration switch 1 (002-2) on gives this option; control cards are read (004 through 010) and the sort branches (010-1) to load part 2 of Phase I (053).

READING AND CHECKING CONTROL CARDS

The first task is to read the control cards. A card is read (004); if it is the first card, the program branches (005) to determine whether there will be a third card for input labels and a fourth card for output labels. If one or both types of labeling are not used, the card reading loop is set for one or two fewer cards (006, 007, 008, 009). The program loops (010) until the last control card is read. Then the user is given an opportunity to exit at AASSIGN (011) for added coding.

Several housekeeping tasks follow (012). Electronic switches used by Phase I are turned off, and temporary and permanent storage areas are zeroed. Switches are set in accordance with whether the user has specified a two-area or three-area system for Phase I, or has left it for Sort 90 to decide (013).

The following 800 or so locations contain instructions (014, 015, 016, 016-1) which take the control information from the locations into which control cards were read, enter it into the appropriate locations of the communications block, and initialize many parts of the running program. Each bit of information is also checked as it is processed. If it falls outside of the range permitted the sort, an error message is typed, and the program comes to unconditional HALT 1001 or, in some cases, to other halts. Usually the user must repunch his control cards and begin again. The sort checks for SPOOL, sets up for the proper record form and length, takes care of options on record counts and hash totals, makes provision for editing, sets up a table for tape allocation, prepares for the specified tape labeling, checks for the overlapping of control fields, and generates a table of control data segments in the communications block.

GENERATION OF COMPARE ROUTINE

With the table of control data segments ready, Phase I assignment generates (017) the compare routine which is used by the scan, the internal merge, and the stepdown check. The number of comparisions per record will depend upon the number of control data segments, of which there may be no more than 16. Four instructions are needed for each segment compared, so with a final unconditional branch instruction the length of the compare routine varies from five to 65 instructions. The compare routine is located at the end of the running program; when it is short, more storage is available for the records to be sorted.

DECISION FOR TWO- OR THREE-AREA I-O

Assignment's next task is to compute the size of the G's and to decide, unless the user has already done so, whether a two-area system must be selected (018). Otherwise, the switches in the computation routine depend (021-1, 021-2) upon ACOMPSWA (019) and ACOMPSWB (020), which were set earlier (013). Assignment then enters (022) one of two subroutines, identical in terms of their major logic but differing in that one handles form 3 records and the other form 1 and 2 records. Their logic will be discussed with reference to the form 1 and 2 routine.

1. When the user specifies a two-area system, assignment branches through the Y branch of AAREASW (023), computes the G size for a two-area system (024), computes the size of the sort blocking (026), and drops through the N branches of swx (027) and SWM (033).

2. If the user specifies a three-area system, the sequence is the same except that it substitutes the G size computation for a three-area system (025).

3. If the decision is left to Sort 90, the sequence is as follows: Assignment computes the G size for a three-area system (025), calculates the sort blocking (026), and branches A at swx (027). If process time is less than either reading or writing time (028), a three-area system is chosen, and a message indicating this is given (034). If not, assignment sets swx to B (029), computes G size for a two-area system (024), selects a new sort blocking (026), and branches B at swx (027). If process time is greater than writing time added to reading time (030), then the two-area system is chosen and a message is given to indicate that a two-area system was chosen because of timing considerations (035). If processing time does not exceed combined tape times, the subroutine sets swx to N (031) and SWM to T (032), recomputes G size for a three-area system (025), selects sort blocking (026), and drops through the N branch of swx (027) and the T branch of SWM (033) to issue a message (034) which indicates the choice of three areas.

If computation of G size (025) for a three-area system indicates insufficient storage available, assignment comes to HALT 1052. When START is depressed, the program will attempt to use a two-area system (024). If this is successful, a message indicates the choice of a two-area system because of space considerations. If not, or whenever the user specifies a two-area system and storage space is lacking, the sort comes to uncon-

ditional HALT 1053. (Chart 2A does not indicate these program steps, which arise when space limitations become a factor.)

CALCULATION OF G SIZE AND SORT BLOCKING

One part (026) of the subroutine being discussed (022-035) computes the blocking (the number of records per block) to be used internally by Sort 90. The size of the block is limited by the storage space available in each of the phases; this space, in turn, varies according to the length of the running program, the extent of added programming such as editing and SPOOL, and the order of merge.

1. The number of blocks which must be in storage is 2(M+1) for Phase II and 2M blocks plus two output blocks for Phase III. From this consideration alone, block size diminishes sharply as the order of merge is increased.
2. The lengths of the running programs in all three phases vary according to the length of the comparison routines and, in Phases II and III, according to the number of comparison routines and file schedulers. The number of the latter increases with the order of merge.
3. Any phase in which the user adds routines for summarizing, editing, and the like will have less room for records.
4. Running SPOOL diminishes the available space.
5. In addition, the user may set a limit to block size by modifying location ABLOCKMAX. (See "Modifications" in J28-6096.)

For form 1 and 2 records, the blocking which the sort will select will be the smallest of the following: G (one group of records for Phase I), the maximum possible blocking for Phase II, the maximum possible blocking for Phase III, or ABLOCKMAX divided by record length. Note that G is made an even multiple of the input blocking, but is not necessarily an equal multiple of the internal blocking. This means that the input of the first pass of Phase II will typically contain a short block for every G processed in Phase I, so Sort 90 is designed to handle routinely short blocks from this source or from any deletion routines. For form 3 records, the block-size is measured in words rather than records, and the size selected will be the smallest of the following: the number of words in G, the maximum possible block-size for Phase II, the maximum possible block-size for Phase III, or ABLOCKMAX.

When both the blocking and G size have been computed, a message is typed listing these quantities (049). For forms 1 and 2, these are given as the number of *records* per block and per G, while for the variable-length form 3 records, these are given as the number of *words* per block and per G. Now that G is known, the routine to generate RDW's for Phase I can be initialized (050), and the storage limits are inserted (051) in the assignment routine which will be used to zero upper storage (069) during part 2 of assignment. ASSIGNLINK (052) gives the user the option of adding programming before part 2 is loaded (053). Before calling part 2, however, the sort zeroes upper storage (052-1). This eliminates the possibility of invalid alphameric characters when the restart routine is written out during the last section of assignment.

PART 2

Part 2 begins with a check for RESTART (054). If the Phase I constant is not in APHASEKEY, the sort branches to load Phase II (055). Otherwise, assignment continues. If no SPOOL is to accompany the sort, it eliminates the SPOOL test from the channel schedulers (056). The sort saves the OPENPROC and LABELINF entries from the output DTF, since they will be altered by the routine which assigns and writes the restart routine; these entries will be needed when the DTF is reset to OPEN all the output tapes except the one containing the restart routine (056). To save time, Sort 90 disables the IOCS subroutine which causes header labels to be typed before the labels are processed in the end-of-reel routine and OPEN (057). It modifies the OPEN routine so that the signs of RDW's in the RDW lists will not be set (058) and sets the output DTF for the channel and unit of the last output tape (059). The restart routine is assigned according to the DCHPT, which the sort set earlier according to the user's specifications. When this assignment is completed, the restart routine is written on the last output tape (059).

The sort saves the file serial number from the output label area for input header checking in Phase II (060) and opens the first output tape (061). Since all output tapes are not yet OPEN (061-1), it sets the output DTF for the channel and unit of the next output tape; it restores the OPENPROC and LABELINF entries of the DTF (061-2). Assignment opens the next output tape (061) and checks to see if it is the last (061-1). If not, the loop continues. Whenever the check reveals that all the output tapes are OPEN, the sort saves the output file creation date for input header checking in Phase II, and sets the input DTF for the first input channel and unit (062). It saves the first input reel sequence number for the possibility of a restart during the end-of-reel routine for this reel (063).

The sort opens the first input tape (064) and restores the OPEN routine's capacity to attach signs to the RDW's (065) in case the user wishes to OPEN any

additional tapes through exit AADDASSIGN (068). If there is more than one input tape, ALT1TAPE and ALT2TAPE are set with the next input unit number to permit the switching of input tapes when the first tape is at end-of-reel (066). Since Phase I does much of the scheduling of its own input-output functions, several modifications must be made to the file schedulers (067). For example, assignment modifies both file schedulers so that when the availability switch is encountered in the ON condition, the program branches to sort coding. Several parts of the sort input-output scheduler are also assigned (067).

The user may exit to added coding at AADDASSIGN (068); this is the place to OPEN any additional files because the OPEN routine is available. Upper storage is zeroed as an aid in checking errors (069), and for form 1 and 2 records the RDW's for each G are generated (070). Assignment initializes the reel-by-reel record and hash counters (070-1) and, in preparation for Phase I's special restart option, writes segment marks on each of the output tapes (070-2). HALT 1108 occurs if error occurs in writing the segment mark. Pressing START causes a backspace, a skip forward, and another attempt to write the segment mark. After a segment mark is written on each output tape, the sort branches to its own I-O scheduler to prepare for the first read (138).

Exits and Modifications

The user may modify Sort 90 by altering constants or by adding program steps. A detailed discussion of these changes and their purposes is given under "Modifications" in J28-6096. For added programming, the user assembles the necessary coding and overlays recommended exit points with branch instructions to the new routines. The added coding should include return branches to recommended reentry points. For modifications which require the reading or writing of additional tape files during the sort, the user must assemble (for each added file) a main program modification consisting of a DTF entry, a file scheduler, a label DC information entry, a DA for record storage areas, the requisite IOCS macro-instructions (including OPEN and CLOSE), the required processing instructions, and the required linkages from and into the sort program.

The major functions for which exits are available in Phase I may be summarized as follows:

1. The user may exit during assignment to execute added assignment routines such as opening additional tape files (068). The exits are AASSIGN (011), ASSIGNLINK (052), AADDASSIGN (068).
2. During the scan the user may exit to do added processing on a record-to-record basis. Deletion of selected records is a possible function of such processing. Additional files may be read or written at this time. The exit is ARECDBRNCH (073).
3. During the input end-of-reel routine the user may exit for his own end-of-reel routines such as processing additional trailer labels. The exits are AMORETRLR (188) and AENDOFREEL (194-1). These replace the IOCSEX6, IOCSEOFEX, and IOCSEOREX used by Sort 90. The other IOCS input end-of-reel exits are available to be used in the normal way, if desired.
4. The end-of-file routine provides exits for end-of-phase processing such as closing any additional input files. The exits are AWINDUP (198) and AENDPHASE1 (199).

Besides exiting to added coding, the user may modify Sort 90 by overlaying certain constants. The detailed instructions are given under "Modifications" in J28-6096, but the following list summarizes the modifications possible in Phase I:

1. The routine in Phase I assignment which computes G size and, if desired, selects between a two- and three-area system of input-output processing can be modified. Its constants can be altered to improve its accuracy in a particular application. For example, programming added by the user may increase significantly the per-record execution time. Though it normally assumes no added execution time, the computation routine can be modified easily.
2. Location ABLOCKMAX can be overlaid with a number to limit the maximum block-size employed internally by the sort.
3. To change the unit from which control card information is taken, the user may modify one of several constants. By proper modification the user may cause this information to be taken from another unit record device, a tape unit, or a storage location instead of from a 7500 Card Reader on Synchronizer 1.
4. The user may alter the IOCS treatment of the input file by changing its DTF entry. The DC label information entry can also be changed to modify IOCS treatment of the input file tape labels.

Storage Maps

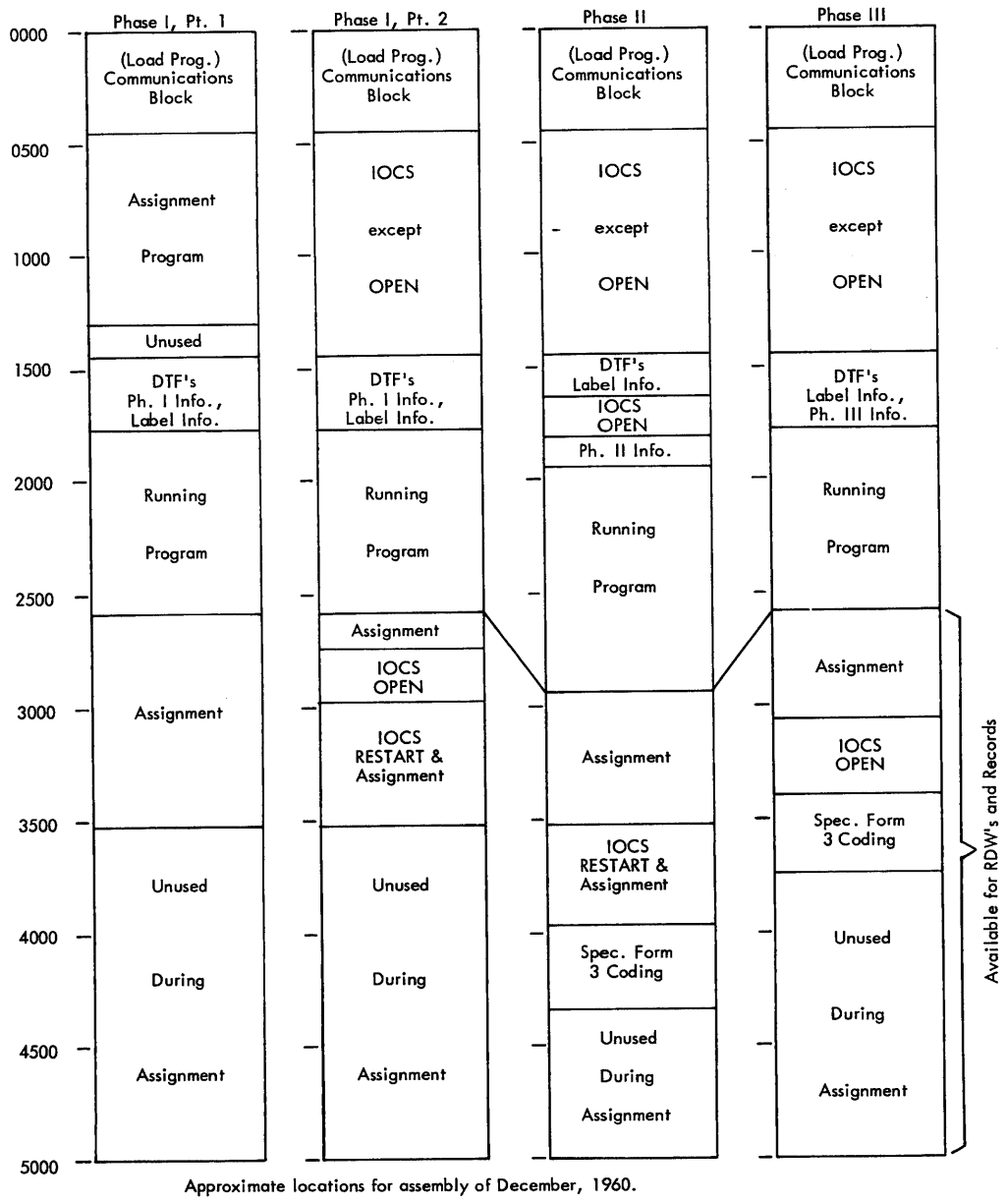
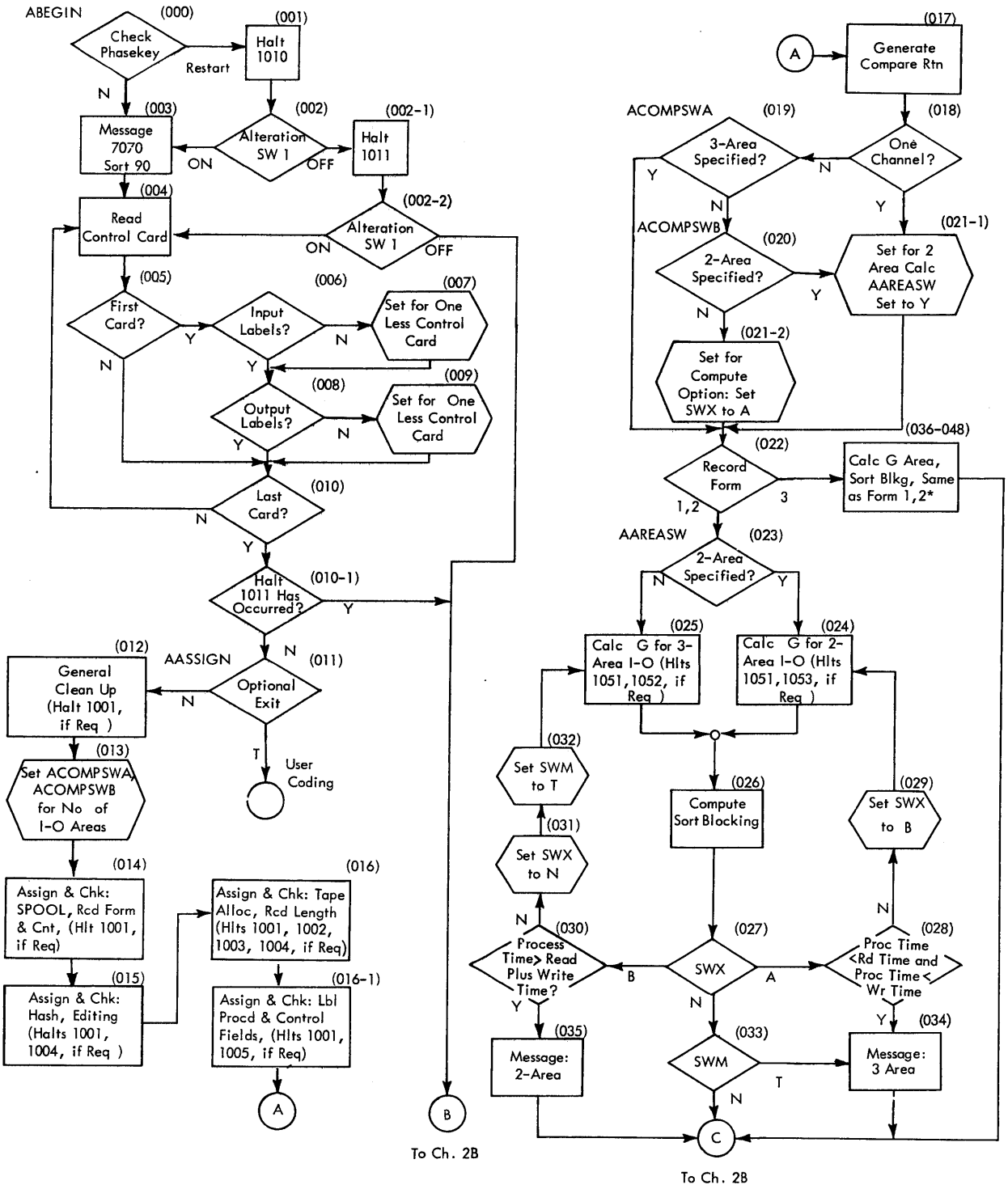


Figure 5. Storage Maps



*Halts 1054, 1051, 1052, 1053 if required

Chart 2A. Phase I; Assignment Routine 1

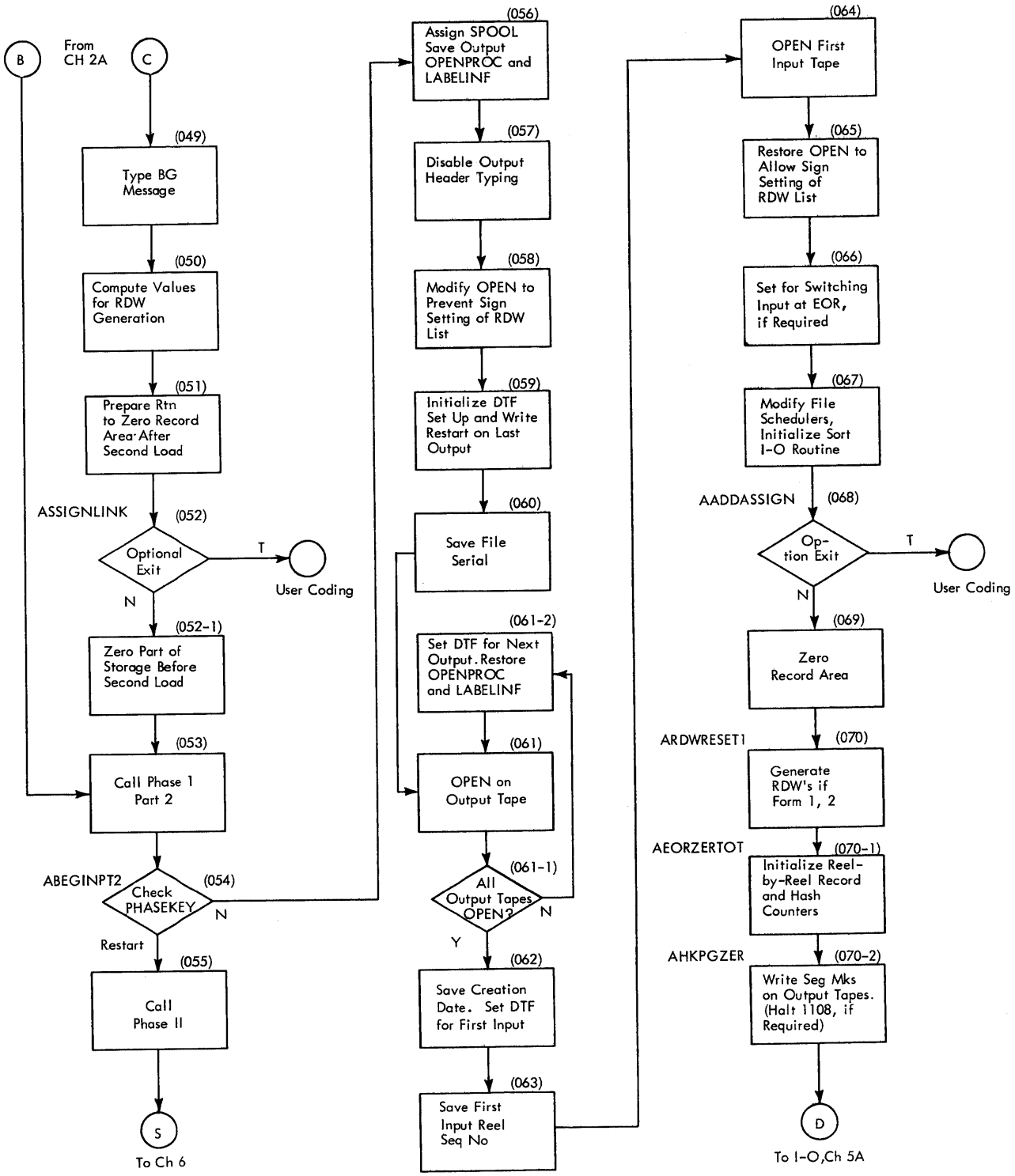


Chart 2B. Phase I; Assignment Routine 2

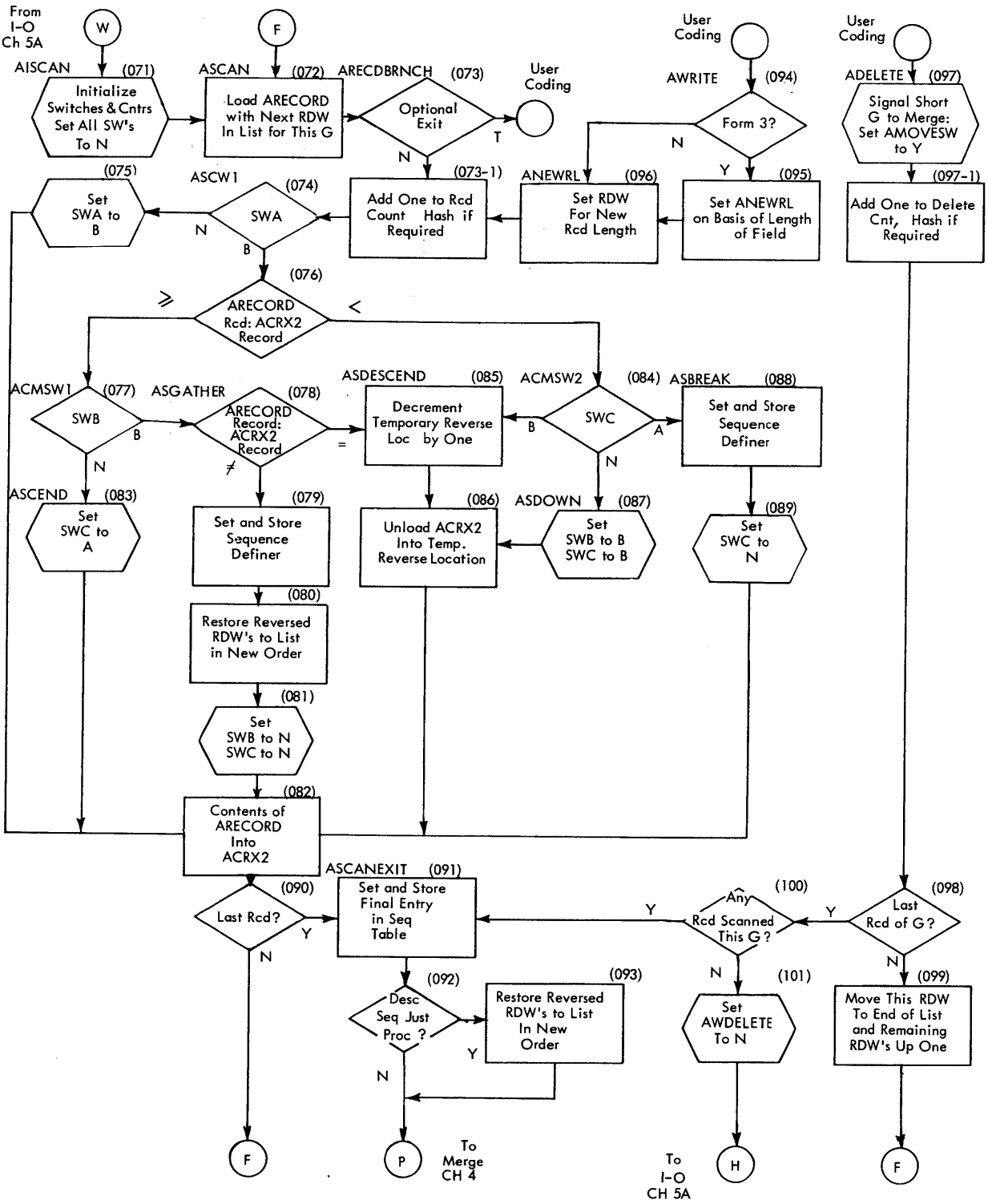
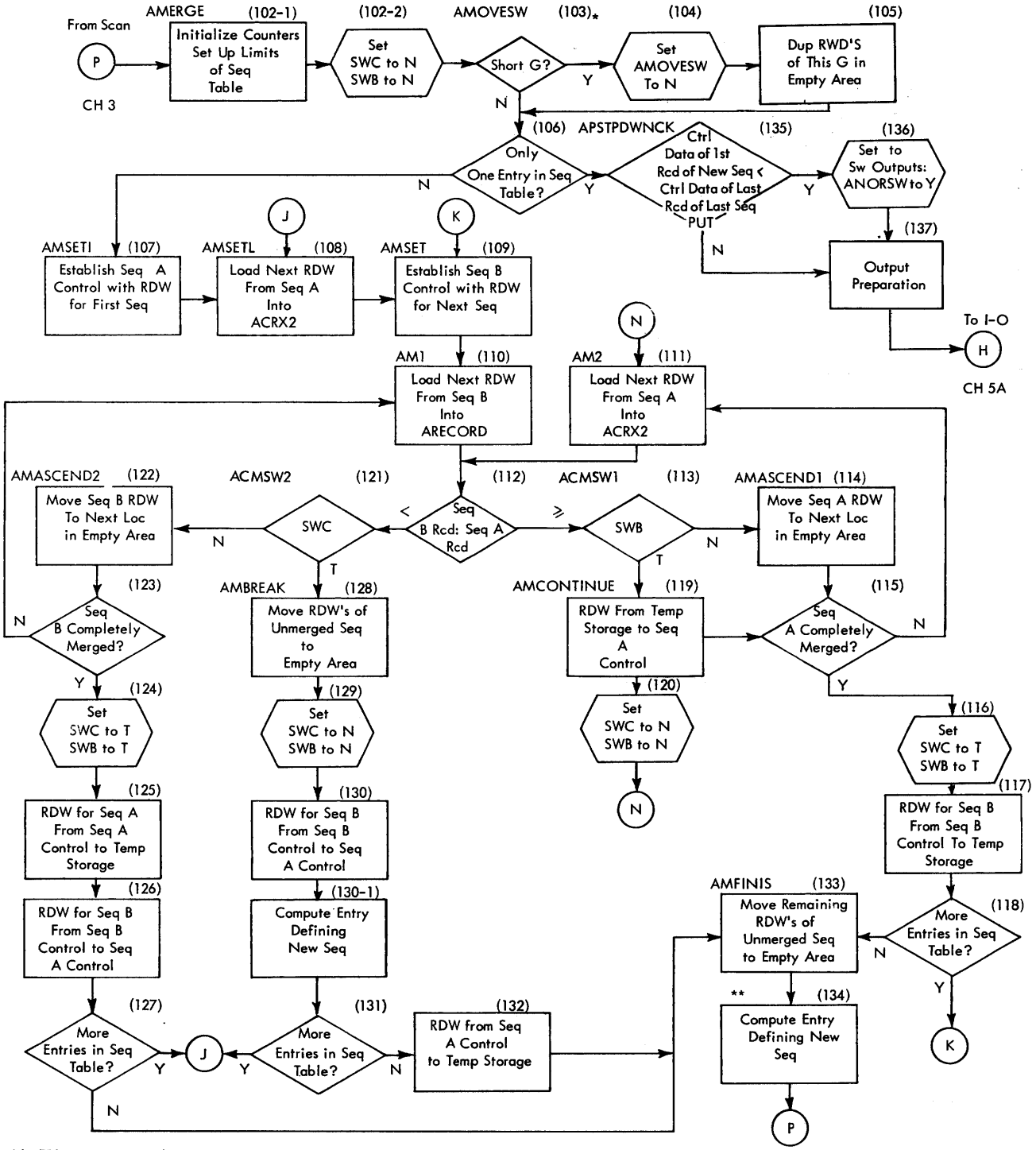


Chart 3. Phase I; Internal Sort, Scan

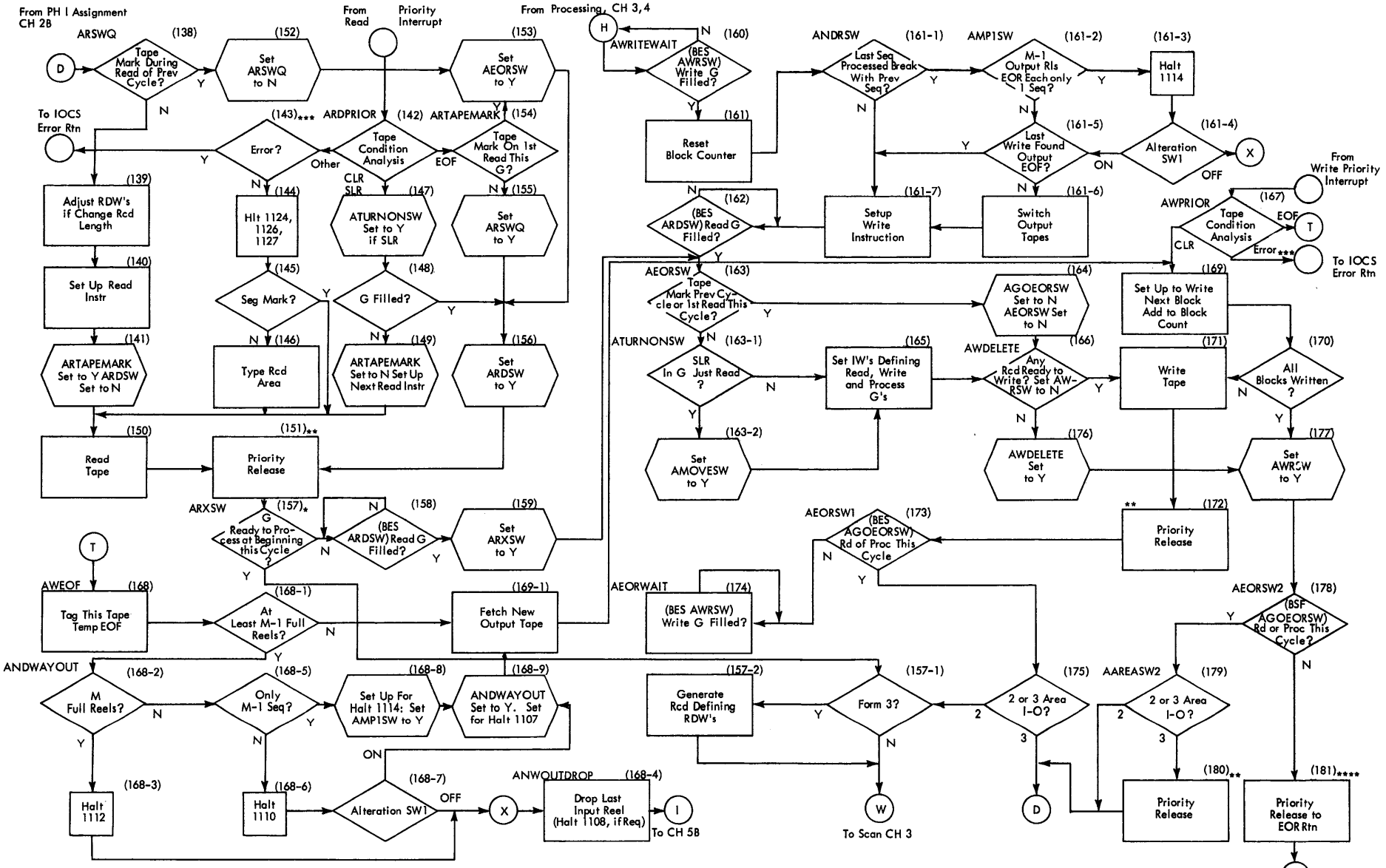


NOTES: * Set to Y by:

1. One or more SLR conditions on the series of reads which filled this G, or
2. A tape mark read before this G was completely filled with records, or
3. One or more records deleted during the SCAN preceding this MERGE.

** Computing the entry defining the new sequence includes setting up the reversing of roles of the two RDW areas involved; on the next merge pass, the area containing the "current" RDW's becomes the empty area into which RDW's are moved, and the formerly empty area now contains the "current list of RDW's.

Chart 4. Phase I; Internal Sort, Merge



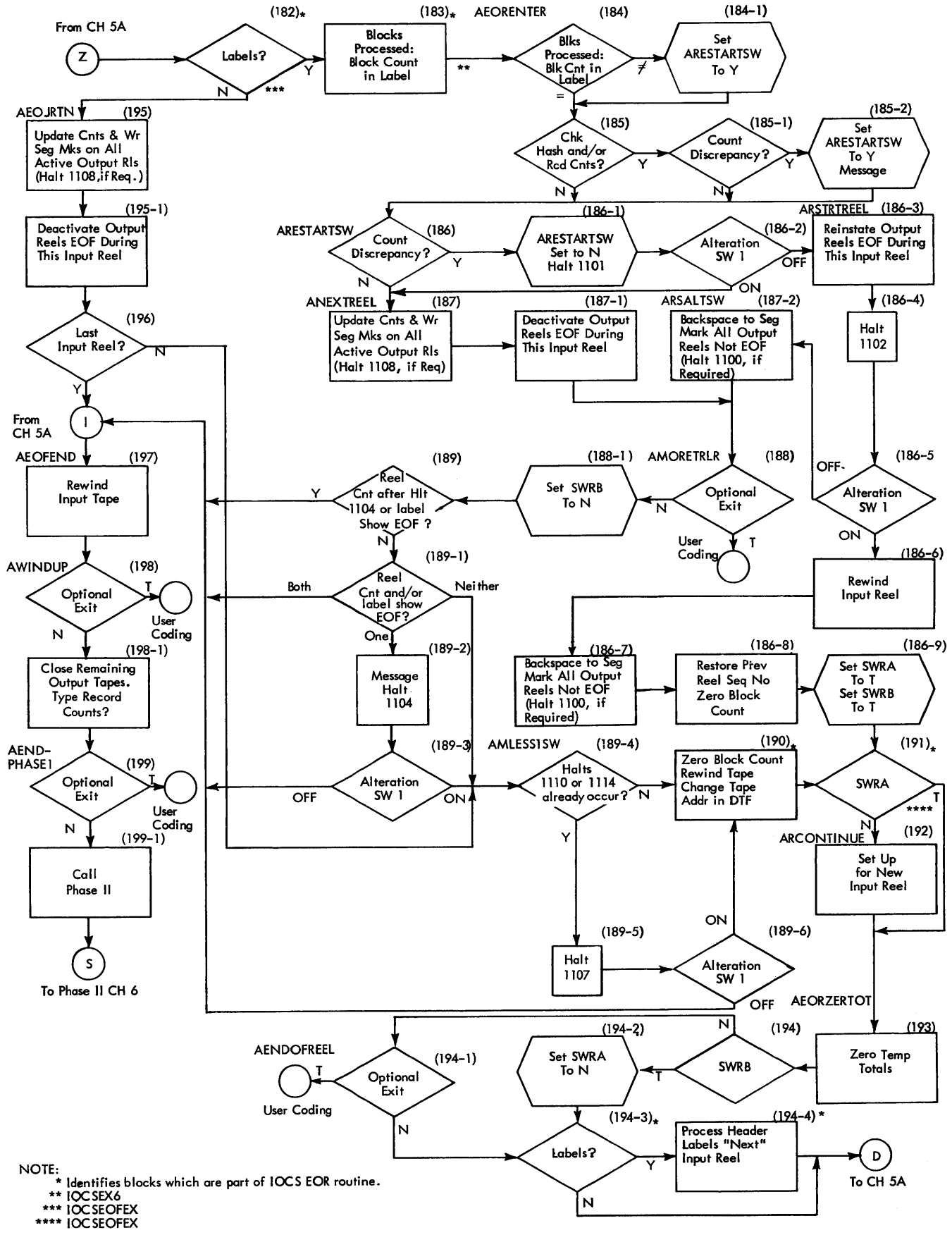
NOTES: * Set to "N" during first G of an input reel.

** Acts as a priority release instruction if in priority mode; otherwise, the program continues along the flow indicated. Actually handled by IOCS schedulers.

*** The IOCS error routine returns to the priority condition code analysis after processing the error record.

**** If in the priority mode, a priority release to the end-of-reel routine is executed; otherwise, a branch to the end-of-reel routine is executed.

Chart 5A. Phase I; Input-Output Scheduling



NOTE:
* Identifies blocks which are part of IOCS EOR routine.
** IOCSEX6
*** IOCSEOFEX
**** IOCSEOFEX

Chart 5B. Phase I; Input End of Reel and End of File

Phase II

Phase II of Sort 90 is a multipass routine which merges the sequences provided by M input tapes into longer sequences and writes them out on M output tapes. ("M" equals the order of merge.) The M input tapes on the first pass are the output tapes of Phase I; each contains a number of sequences formed by the internal sort of Phase I. Phase II is started by taking one sequence from each of the input tapes. These M sequences are merged into one sequence which is written on one of the output tapes. A second sequence is taken from each input tape, and these sequences are merged into a second sequence on a second output tape. This process is repeated, with the output rotated from tape to tape, until all sequences have been merged and the pass is complete. Typically each output tape will receive at least one sequence. As long as any tape receives more than one sequence, at least one more pass will be required. In succeeding passes the input tapes of the previous pass become the output tapes for the current pass and the former output tapes become the input tapes. (See Chart 1B.) It is possible that none of the output tapes of Phase I contain more than one sequence; if so, Phase II processing will be completely bypassed.

Phase II, like Phase I, has both an assignment program and a running program. However, since some of the initialization must be repeated every pass (e.g., opening and closing files), the running program can be further subdivided into the beginning-of-pass procedures, the merge, and the end-of-pass procedures. A memory map shows the relative locations of the assignment and running programs as they are loaded into storage. See Figure 5.

When not otherwise specified, the collating sequence is assumed to be ascending.

Running Program

General Description

To carry out the merging successfully, the running program must not only order the records from the current set of input sequences, but must also determine when the current sequence from any given tape is exhausted, prepare records for processing and writing, and regulate the tape operations. The following sections discuss the general techniques by which Phase II accomplishes these tasks.

COMPARING AND RANKING

The merge begins when the lowest record from the first sequence on each of M input tapes is available. Immediately it is faced with the question of which record is to be written first. Then, when the lowest is PUT, it must GET the second record from the same input tape and again decide which record is lowest. To decide which of the next available records, one from each input tape, is to be merged, Sort 90 uses a compare network and ranking. (The ranking is illustrated by the lower half of Figure 8.)

The ranking is a series of index words, one pair for each input file. One word of each pair will contain the RDW of a current record from each file and the other will carry a branch instruction to identify the file from which the record came. The series of index words is called a ranking because they are loaded so that the RDW's of the current set of records, one from each input file, will be in an order which ranks the records according to the collating sequence. The next record to be merged will be the one whose RDW is in the lowest position in the ranking.

When the lowest record is PUT, the RDW for the next record from the same input sequence is obtained. The control data of the record are compared with those of as many of the other current records as necessary, beginning with the lowest. The sort continues through the compare network until one of the already-ranked records is found to be equal to or greater than the new record. When this occurs, the RDW's and branches of any lower records are moved down the ranking, and the RDW and branch of the new record are inserted in their proper positions. The lowest record, which may be the newest one or one of the others, is PUT—unless the user wishes to delete it or summarize.

The process is repeated until all records from the current sequence of each input file have been merged. When this happens, the next set of sequences, one from each input file, will have their records ranked and then merged into a second sequence. This cycle is repeated until all input files are exhausted and the pass is completed.

STEPDOWN CHECK

Since each input tape typically contains more than one sequence, the program must discover when the one currently being merged is exhausted. The program identifies the exhaustion of a current input sequence

and the start of another by comparing the first record of each new input block with the last record PUT. If the control data of the new record are lower (a stepdown) a new sequence is indicated. The RDW's and branches of the first records of stepdown files are entered into the ranking above those of any records of the files still merging into the current output sequence. The RDW and branch of the first record of a new stepdown file is compared with any other stepdown records represented in the ranking. It is entered in its proper position in the ranking—after the RDW's and branches below it are moved down one pair. Switches are set which restrict new records from files not yet stepdown to comparisons with the currently ranked records of other non-stepdown files; thus the merge continues without the stepdown files. When a stepdown has been encountered in each input file, all of the input files become active again for merging into a new sequence. This sequence will be written on the next available tape.

RDW INTERCHANGE

Records of forms 1 and 2 are readied for processing and writing without being moved in storage. As in Phase I, the merge is accomplished by the movement of RDW's, but in Phase II the method is one of RDW interchange or exchange (PUTX). A GET and the succeeding compare routines cause the RDW of a record to enter the ranking as discussed above. When a record is PUT, its RDW is transferred to the output RDW list, but before it is entered, the RDW from the location it is to occupy is temporarily stored. When the program returns to GET another record from the same input file, the RDW is unloaded from temporary storage into the location in the input list of the RDW for the record just PUT; this completes the exchange.

The technique of interchange has interesting consequences for the location of records and RDW's in storage. As sorting begins in any pass of Phase II, upper storage will contain the record areas and any added programming. (See Figure 6.) The record area can be divided into 2 (M+1) blocks, each identified by an RDW list.

At the start of processing, the records of each block are read into contiguous storage locations. (See Figure 7A.) Records are compared and ranked. When the ranking is full, the RDW of the lowest record enters the output list, initiating an interchange. The interchange is completed with the output RDW entering the input list. Another record is ranked, and the RDW of the lowest again enters the output list. When the first output list is full, the block it controls will be written out while another is filled. When an input list has been completely interchanged, a new block of records will be read, but this time, instead of going into contiguous areas of storage as at the start, the records will be scattered into the locations represented by the RDW's which have been moved from the output list into this input list. On the second read into a block these may represent mainly RDW's originally in the two output blocks. But on later reads into a given input block, RDW's which started in any of the input or output lists may be present. As a result, the latest block of records read from any tape is very likely scattered over the whole area reserved for input and output blocks rather than being read into contiguous areas as during the first GET. Similarly, output blocks consist of records scattered throughout this area. The scattering of records has no influence on the outcome of the sort operation itself, since the important factor is contiguous lists of RDW's. Knowledge of this feature, however, may aid a user who examines a storage print made during Phase II. (See Figure 7B.)

If processing should be interrupted, a record at any given location in storage may be in any one of the following conditions:

Not yet written,	its RDW in an input list
Not yet written,	its RDW in an output list
Being written,	its RDW in an output list
Already written,	its RDW in an output list
Already written,	its RDW in an input list

NON-ENDING SORT

The maximum number of records which Sort 90 may safely sort is ordinarily the number which occupy M−1 full reels when the records are blocked according to the sort blocking factor. If the user attempts to force the sort to merge a greater number of records, he may encounter a non-ending sort in Phase II. The sort cannot end with a sequenced file whenever a long sequence completely fills the first M−1 reels and shares the other reel with one or more shorter sequences. Further passes will not change the order of the records at all. On another pass the first sequence on each tape (the only sequence on M−1 of the tapes) merges again with the other first sequences, but these already form

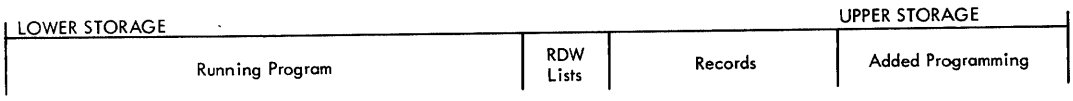


Figure 6. Storage Allocation, Phase II

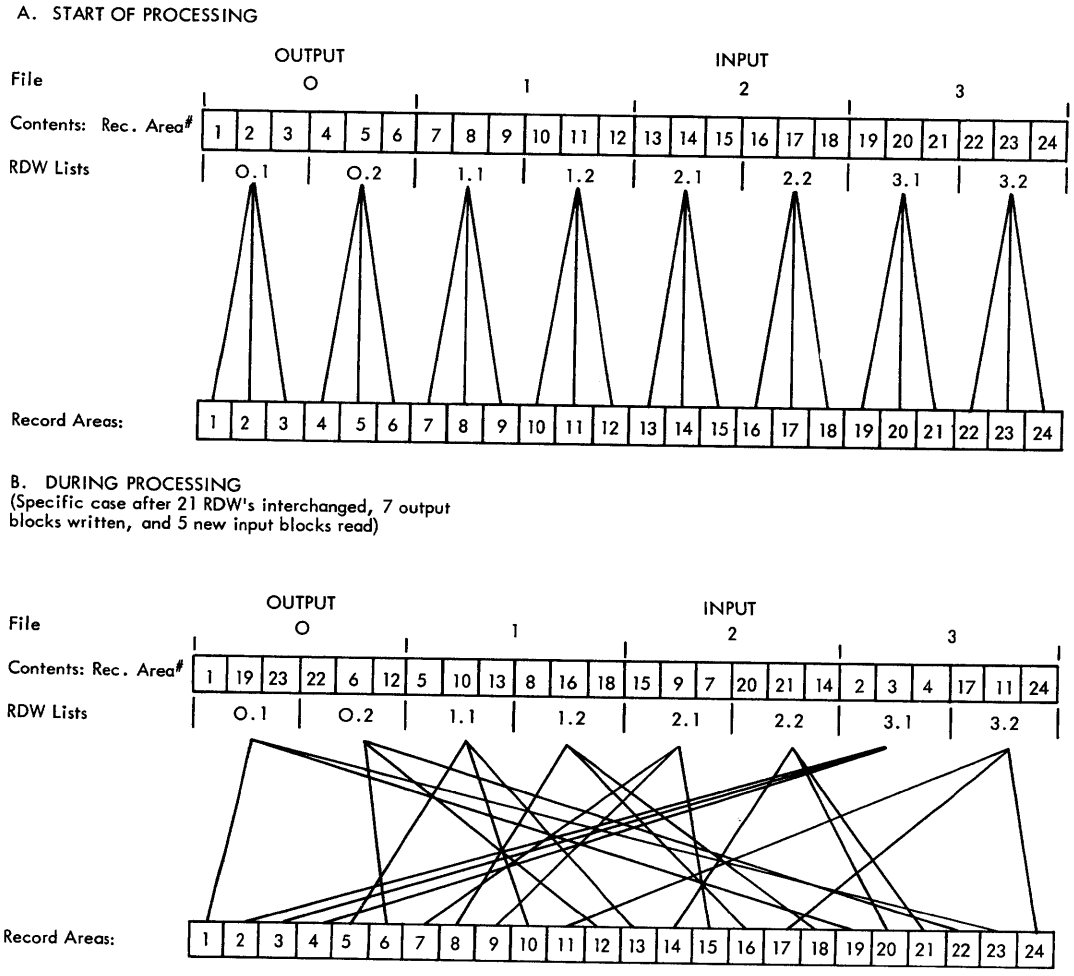


Figure 7. RDW and Record Areas, Three-Way Merge, Blocking Factor of Three

one long sequence, so there is no change in order. There are no sequences left on other input tapes to merge with the remaining sequences on the last tape, so these latter are written just as they are. Since there is no reduction in sequences, HALT 2000 occurs. If the user elects to continue, no further change normally takes place. One possible exception may occur when summarizing or deletion did not take place on the pass when the above condition first occurred but is expected on some succeeding pass. For any chance of the successful completion of the sort, any such future reduction in the number of records would have to shorten the long sequence so that it occupies no more than M−1 full reels.

HALT 2000 in the beginning-of-pass procedures stops the sort whenever the number of sequences has not been reduced over the preceding pass. However, the user is warned of the possibilities of an unending sort much earlier than this, and he can avoid them entirely, if he uses no more than M−1 reels of input—with a blocking factor no greater than that of the sort—or if he makes sure that no output reel from Phase I contains more than one sequence. In the latter case Phase II would be bypassed completely. The sort provides the following halts and a choice of action in Phase I:

1. When M−1 reels have been filled: HALT 1110.
2. If each of the first M−1 reels contains one sequence, when the last reel encounters a sequence break (step down): HALT 1114.

If the user elects to drop the current input reel after these halts, the sort may proceed to a successful completion (definitely so after HALT 1114). If, instead, he decides to process more input records, the sort will most likely not succeed unless one or more of the Phase I output tapes are short, or unless heavy summarizing will be performed in early passes of Phase II. If the user elects to enter more input records, he

is also notified at HALT 1107 when the current input reel is exhausted and/or at HALT 1112 when all M output reels are filled.

Example of Phase II Processing

Figure 8 provides an example to illustrate processing in Phase II of Sort 90. The figure is divided into three rows of blocks; the top two rows represent ten RDW lists; the bottom row represents the ranking, at six successive steps. Below the rankings are listed the comparisons which were made to insert the newest RDW and branch into the ranking during each interval, together with the result of the last comparison.

RDW LISTS

The example illustrates Phase II processing for a four-way merge, where the blocking factor is three, where internal blocking is not an even multiple of the G of Phase 1, and where the control data consist of a two-digit field. Therefore, ten RDW lists are represented, one pair for the output and a pair for each of the four input files. One member of each pair is available for writing and reading, while the other is being processed. Each RDW list contains three words, but the figure shows six lines, with the "a" lines indicating the contents of the three words before this list is processed and the "b" lines containing any RDW which, through interchange, overlays the contents represented by "a." Words in the current input lists which were entered in the write list prior to step 1 are left blank in order to make clearer the condition at step 1. The right-hand columns of each of the sixteen blocks represent the control data of the records defined by the RDW's in the lists before these records have been written. For this reason no control data are represented for the RDW's initially in the output blocks. An asterisk (*) in the control data column indicates a record area and an RDW not currently used because a short block is present.

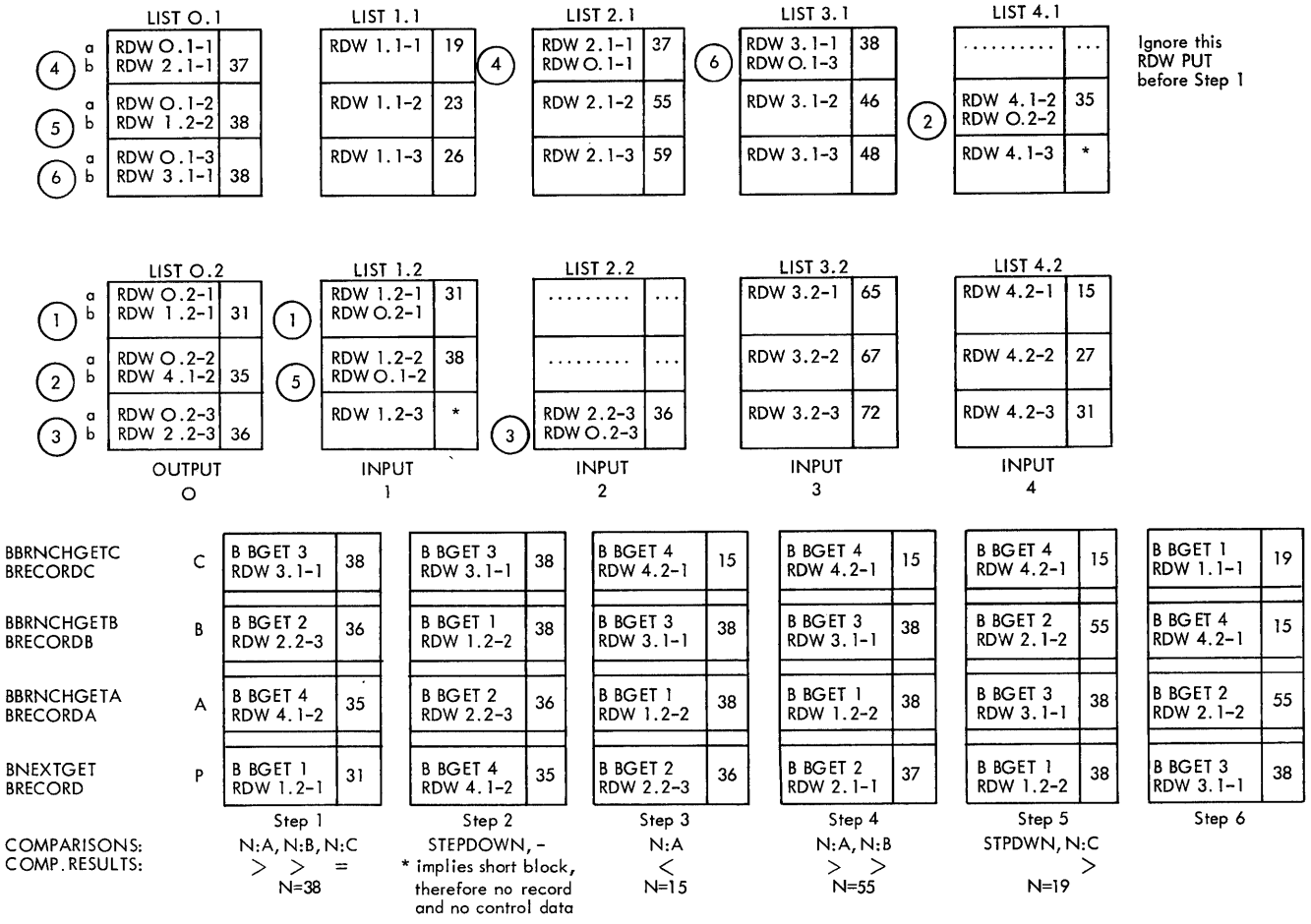


Figure 8. Example of Phase II Processing

THE RANKING

To the left of the blocks representing the ranking are the four pairs of index words used for the ranking in a four-way merge. The four letters may be used as brief symbols for the records referenced by the RDW at each level during a single interval. (These letters are also used several times to refer to the pair of index words which represent this record in the ranking, but the context makes very clear when this is the case.) N is used to symbolize the new record before it is ranked. In order to insert it in the ranking it will be compared (N:A, ...) with as many as necessary of the current records in the ranking except P. P at that stage represents the record last PUT in the output list, so that BNEXTGET and BRECORD do not reference a current record and are in effect "empty." N and its input location are defined by a comparable pair of index words, BNEWRECORD and BBRNCHGETN. These unload the necessary information at the proper level in the ranking after that level is determined by the comparisons. It may be noted that the order of the merge determines how many levels of the ranking are necessary and how many of these index words will be used. For a two-way merge, only the A pair and lower would be needed, while for a five-way merge, BBRNCHGETD and BRECORDD would be added to those illustrated. Note a related fact: the number of comparison routines needed will vary with the order of the merge, and, as discussed elsewhere, only the number needed are actually generated by the assignment program. In Figure 8 the comparisons listed under a step are those used on records of that ranking to obtain the ranking listed for the next step.

MERGING THE RECORDS

Basically, Figure 8 shows a series of comparisons, rankings, RDW interchanges, and stepdowns. To aid in identifying the RDW's exchanged at a given time, a circled number has been placed beside the RDW's which are to be exchanged. This number is the same as the number of the step.

Step 1: At step 1 the "a" lines of the RDW lists are as given. All "b" lines are empty. An RDW from each input list has been ranked, and RDW 1.2-1 referring to the lowest record is entered in output list 0.2, which is now ready for processing. (The records represented by the "a" lines of list 0.1 are currently being written.) Before RDW 1.2-1 overlays RDW 0.2-1, the latter is moved to a temporary location. The program comes to instruction BNEXTGET. Since its contents, symbolized by BBGET1, is a branch to instructions for the first pair of input RDW lists, the program unloads RDW 0.2-1 from temporary storage into list 1.2, where it overlays RDW 1.2-1. This completes the RDW interchange. The sort GETS the next RDW from this list, RDW 1.2-2, by loading it into BNEWRECORD, and sets BBRNCHGETN to BBGET1. The record, N, referenced by RDW 1.2-2 is compared with A. Since N's control data (38) are larger than A's (35), N is compared with B. Since N's control data are also larger than B's (36), N is compared with C. N's control data are equal to C's, so the contents of the A pair of index words are moved to the P pair; the contents of B pair are moved to A pair, and the contents of the N pair are moved to the B pair. This gives the sequence represented by the ranking at step 2.

Step 2: RDW 4.1-2 is now in BRECORD. It is PUT into the next available location of output list 0.2, and a sequence of events follows which is similar to that described for step 1. RDW 0.2-2 is temporarily stored, and RDW 4.1-2 overlays it in list 0.2. RDW 0.2-2 is unloaded into list 4.1, the list currently being processed for the fourth input area, because the branch instruction of BNEXTGET is BBGET4. Since list 4.1 currently holds a short block, this is its last entry. Therefore, list 4.1 is readied to control the reading of more records, and list 4.2 when ready is used for processing. RDW 4.2-1 is loaded into BNEWRECORD and BBRNCHGETN is set to BBGET4. N is now the first record in a block, so a stepdown check is made. The stepdown check reveals that list 4.2 references a new sequence on this input tape; N's control data (15) are lower than those (35) of the last record PUT. Since this is the first input tape which is stepdown, switches are set to prevent records which may yet enter the current output sequence from being compared with C. Then the contents of the A pair of index words are moved to the P pair, B to A, C to B, and N to C. This results in the ranking shown at step 3.

Step 3: The events of step 3 are similar to those of the preceding steps, so only its distinctive characteristics are discussed here. Entering RDW 2.2-3 into list 0.2 fills it, so it is readied for reading. RDW's will be entered into list 0.1 for the next three intervals. After the interchange the processing of input list 2.2 is complete, so it will control the reading of new records while list 2.1 governs the processing of records from the second input tape. This time N is not stepdown, and the sort enters the compare network normally. N is compared with A only, however, since its control data (37) are lower than A's (38). After the single comparison (N:A), N's RDW and branch are entered directly into BRECORD and BNEXTGET, and no others are moved. The ranking is as shown in step 4.

Step 4: RDW 2.1-1 is entered in the output list, and the exchange is completed by unloading RDW 0.1-1 into list 2.1. The control data (55) of N are now larger than

those of the other records ranked for the present output sequence, so its RDW and branch enter BRECORDB and BBRNCHGETB. Because switches were set when the stepdown occurred during step 2, N is not compared with C. The ranking has become as shown in step 5.

Step 5: RDW 1.2-2 is interchanged with RDW 0.1-2. Another short block and stepdown are encountered. For this stepdown, however, the two records for the following sequence must be compared (N:C). The control data (19) of the new record are higher, so the contents of all pairs of index words (A, B, C) are moved down the ranking, and N's RDW and branch are unloaded into pair C. Again switches have been set so that the next record whose control data exceed 55 will be compared with A but never B or C. The ranking is as shown in step 6.

INITIAL FILLING OF THE RANKING

The initial entering of RDW's and branches into the ranking is not shown in Figure 8, but it is accomplished in a way similar to that for stepdown records. When sorting begins, the program loads the first RDW from the first input list into BNEWRECORD and initializes BBRNCHGETN with a branch to this list. The irrelevant contents of pairs A, B, and C are moved down one pair, A to P, B to A, and C to B, and the index words for N are unloaded into pair C. BNEWRECORD and BBRNCHGETN are then loaded with the first RDW and branch for the second input area. This time N is compared with C. If N has higher control data, A pair is moved to P pair, B to A, C to B, and N to C. If not, A pair is moved to P pair, B to A, and N to B. The next GET is for a record from the third input file. N is compared with B, and, if higher, with C. The contents of index words are moved as before until the RDW's and branches rank the records A, B, C in the proper order. A GET is performed for the first record of the fourth input file, the necessary comparisons are made, and the contents of the proper index words moved, so that P, A, B, C are represented in rank order. Normal operation is resumed as P is PUT.

END-OF-REEL

After a GET which reveals an input end-of-reel, the sort closes the input tape, moves the contents of index word pairs A to P, B to A, C to B, and removes C pair from use. After closing the second input tape which becomes EOF, the sort moves A to P, B to A, and removes the B pair from use. The A pair is removed by a third EOF, and a fourth EOF leads to end-of-pass routines.

OTHER FEATURES OF PHASE II

1. After each record is PUT, record counts and hash totals (if required) are taken before the next GET is performed.

2. Provision is made for user exits for editing, summarizing, or deleting records. These optional exits occur after N has entered the ranking and before P is PUT. If a record is deleted or summarized into another, special reentries are provided which bypass the PUT and RDW interchange and allow separate counts of records and hash. Thus P is *not* always PUT.

3. It is evident that the RDW's initially occuring in any list are soon scattered throughout all the lists. This means that the records represented on any one list are likely to be scattered throughout the entire record area.

4. With form 3 records, an RDW is generated for each record, and the comparing and ranking are performed as for form 1 and 2. However, there is no RDW interchange, since each form 3 record is moved to the output area as it is PUT. Because there is no RDW interchange, form 3 records are not scattered. The block of records from a given input file always occupies the same location in storage, and records are located in storage in the same order as on tape.

5. If the user specifies a descending collating sequence, the sort proceeds in the same way except that the branch-if-low instructions in the compare network are changed to branch-if-high, and the branch-if-high instructions are also reversed. P, A, B, C are then in descending order, and P, the highest record, is PUT.

Description of Charts 7 and 8: Merge

The details of the merge in Phase II, which become evident from an examination of the flow chart, largely concern the variety of provisions which must be made for filling the ranking in the first place, for reducing the ranking as input tapes become end-of-file, and for handling stepdowns which are indicated by the first records of new input sequences. The stepdown records must not be added to the output sequence until all tapes are at stepdown.

FIRST GET LOOP

To start the merge the ranking must be filled. When the tapes have been opened, completing the beginning-of-pass procedures, a GET is issued for the first record in the first input block (235). (Since this is initially empty, the program branches to new block routines, the file scheduler, and IOCS to fill and ready this block before processing can continue.) When the first record is available, its RDW is loaded into an index word (BNEWRECORD) used to hold the newest record, and a branch to the GET for this tape is loaded into BBRNCHGETN.

Since this is the first record in the block, the program reaches the routine whose first instruction is

BSTPDWNCHK (240). This is the first GET for this tape, so the sort encounters the instruction to branch to BIGETENTRY (241). An indexed BDX (241 and 242) instruction with its 6-9 portion set according to the order of merge causes the program to branch (245) to the instructions reached by line M−1 where M is the order of merge. For example, if the order of merge is five, 259 and 271 on the diagram represent the instructions performed. If the order of merge is four, 258, 265, and 270 are chosen instead; the instructions represented by locations 266, 259, 271 do not exist, and sw 4 (265) has been modified so that it always remains set to T despite instructions which apparently change its setting. In the move instruction (represented by 271 if M equals 5) the current contents of the positions in the ranking are shifted downward one pair of places, and the contents of BNEWRECORD and BBRNCHGETN enter at the high-order end. What reaches the lowest pair (P) is irrelevant because at BRECEXIT (272) the program branches to GET a record from the next tape area without processing a record. The branch at BRECEXIT was set to AA during beginning-of-pass procedures.

The first record of the next block is obtained and the same sequence of instructions is performed except that the program branches (245) to the instructions reached by line M−2. If the order of the merge is five, these instructions are represented by blocks 258, 265, 266, 270, and 271. The control data of the newest record are compared with those of the first whose RDW and branch are now at the top of the ranking. If the newest record is larger, the branch is to 271; if less than or equal to the first, the branch is to 270. In the former case the contents of BNEWRECORD and BBRNCHGETN enter at the high end of the ranking after all the other pairs have moved down a pair of steps. In the latter case, their contents enter into the second highest pair (C) after the lower pairs have moved; the RDW and branch representing the first record remain in the top pair (D). Then the program branches to GET a record from the next tape area, again without processing a record. This type of loop is repeated until the ranking is full except for the lowest pair of positions, and the last exit used from 245 by the first GET loop is line 1.

FIRST RECORD PUT

After the first GET from the one remaining tape area is complete, the program continues through 239, 240, 241 and the Y branch of 242. Then BSTPDWNCHK (240) is reset (246) to normal operation (N), and BRECEXIT (272) is reset (247, 248, 249) to DD if the user is to summarize, and to BB or CC, if not.

Next, the sort prepares to switch output tapes (251). It resets BSTPDWNSW (244) according to the order of merge and sw 1 (255) to normal (N). Entering the compare network through sw 1, the sort compares the newest record (currently the first record from the last input tape being used) with the lowest-ranking record already obtained. The latter record's RDW and branch now occupy the second lowest pair of positions, A, in the ranking. If the newest record is lower or equal, it is placed (267) in the lowest pair, P (BRECORD and BNEXTGET). If not, and the order of merge is greater than two, the newest record is compared with the second lowest and the same type of decision is made. Wherever the newest record ranks, its RDW and branch are inserted and records moved so that the lowest pair is ready for further processing.

The program branches at BRECEXIT to BB, CC or DD—to CC if the user adds coding here, to DD if he wishes to summarize. After the DD branch, BRECEXIT is changed to an EE branch, since a record now exists against which the next can be compared in the BCMPSEQW (273) routine.

All paths of leaving BRECEXIT rejoin at BPUT (277) where the next record (form 3) is placed in the output block or its RDW (forms 1 and 2) is placed in the output list. If (278) the program is summarizing this pass, or in all cases if it is form 3, the RDW of the lowest record (BRECORD) enters BBASERECRD where it allows comparison with the following record. The record count is increased (280) and, if the user has specified hash totals, the field for hashing in the current record is added to the hash total (281). The first full cycle from GET through PUT is complete.

NORMAL LOOP

The next GET (235) is performed from the area of the record just PUT. The program branches (239) to sw 1 (255) unless records are long and there is only one per block; the compare network is entered normally (260). The comparison and remaining instructions in the loop proceed as in the previous loop except that EE is the branch at BRECEXIT instead of DD when the user has specified summarizing. If summarizing, therefore, the present lowest record (BRECORD) is compared with the last record PUT (273), and the user may cause a branch to added coding, if these two records are equal. If the program branches to user coding from optional exit BSUMBRANCH (275) to summarize or delete, the reentry should be at BSUMMARIZE (281) or at BDELETE (284) where separate counts are kept of records and hash totals.

END-OF-FILE LOOPS

If, on issuing a GET (235) for a record from the area of the record last PUT, it is discovered that the relevant tape is end-of-reel (EOF), the program branches to

CLOSE this file (236). If all input tapes have now become EOF and the merging has been completed for this pass, the program branches (237) to end-of-pass procedures. If not, the entrance (245) to the compare network is modified (at 238). If the order of merge has been five and is now to continue at line 4 with the completed input tape eliminated, an entry (245) through line 4 leads to instructions (259) which set sw 4 (265) to T in such a way that for the remainder of this pass operations to modify sw 4's setting have no effect and the comparision N:D (266) cannot occur. Then all the pairs in the ranking move down a pair of positions (271). In the series of instructions between BRECEXIT (272) and the next GET (235), the new lowest record is treated as in a normal loop. The information loaded into D is irrelevant to the merge and, because sw 4 acts as a barrier, will never be considered.

When the next tape is discovered to be EOF as an attempt is made to GET a record from its block, the same steps are followed, except that the entry (245) to the compare network is by a line numbered one lower. This number, incidentally, represents the order of merge of the remaining tapes which are not EOF. If the original order of merge had been five and was now to become three, sw 3 (263) is set to T (258) for the remainder of the pass and the instruction (258) to modify sw 4 has no effect. sw 4 (265) then transfers the program to instructions (270) which move the three relevant pairs in the ranking down a pair. Because the same coding happens to be used for other loops, these instructions also move the contents of BNEWRECORD and BBRNCHGETN to the next higher pair, C. This pair ceases to concern the sort because sw 3 is now a barrier.

This loop is repeated with the following entries (245) to the compare network through lines M−1 to 1. When all tapes are EOF, the program branches (237) to the end-of-pass procedures (212).

END-OF-PASS PROCEDURE

See Chart 6 for this procedure. Index words are used as counters, and at the end of a pass their contents are added to the proper storage locations to give the number of records PUT, deleted, and summarized (212). Then output files are closed (213). If records were summarized or deleted, or if any blocks were dumped as a result of an error condition, the necessary count messages are typed (213). Finally, hash and record counters are checked against the results of the previous phase or pass (214). If there are errors, a message gives the record counts and hash discrepancies, and the program comes to HALT 2199 (215). If discrepancies occur, the user may wish to restart the current pass and may do so using normal IOCS restart procedures. Otherwise, the user may press START to return the sort to the beginning-of-pass procedures (216), the same action as when no discrepancies occur.

STEPDOWN LOOPS

In a normal loop when a GET is performed and the first record in a block is obtained, the sort must check to see whether a new sequence is starting or whether the new block is simply a continuation of the old sequence. This is done in the BSTPDWNCHK routine (243). If the first record in the block is higher than the last record PUT (BRECORD), it is handled normally; the sort enters the compare network through sw 1 (255). If it is equal, the sort bypasses the compare network, places the RDW and branch directly in pair P, and continues normally from BRECEXIT (272). If the new record is lower, the following stepdown loop of instructions is performed: The program branches to BSTEPDOWN (244) and modifies the compare network entry (245). When this is the first tape which is ready to merge its next sequence, the entry is through the line numbered M−1. If the order of merge is four, then the sort branches through line 3 to 258, 265, and 270. The three pairs (A, B, C) representing current records to enter the presently-merging sequence move down the ranking (270) by one pair and the RDW and branch for the new record enter into the highest pair, C. sw 3 is set (258) to prevent records which are to enter into the current sequence from other non-stepdown tapes from being compared with the new record. After the pairs move, there is a new lowest record whose RDW is in BRECORD, and the processing of the current record continues from BRECEXIT as in a normal loop.

Each succeeding GET initiates a normal loop until the next stepdown is encountered. This occurs as the first record of a new block from another tape proves to be lower than the last record PUT. The stepdown causes the program to branch again through 244 and 245 and enter the compare network through the line M−2. If the order of merge is four, instructions represented by blocks 257, 263, 264, and 269 are executed. sw 3 is returned to N and sw 2 is set to T. The barrier which existed at sw 3 is changed to one at sw 2. This allows the comparison of records in the current output sequence with each other and records in the next output sequence among themselves, but comparisons between members of the two sequences are prevented. The sort compares the new stepdown record with the earlier one, and ranks them accordingly. If the new one is lower, the pairs in A and B move down the ranking and the new RDW and branch are entered into the third pair, B. If the new record is higher, all con-

tents of the ranking are moved down one pair and the new RDW and branch are entered at the top, C. Then the lowest record, one merging into the current sequence, is processed as in a normal loop starting from BRECEXIT (272).

As other tapes reach the stepdown condition, similar loops are followed. When the last remaining tape has reached this point, the branch at 245 is through line zero. If (252) there is summarizing this pass, BRECEXIT is initialized (253) to DD for processing the first record of the new output sequence. The sort makes the last used RDW in the output block minus, and signals IOCS that the block is ready to be written (254). It sets a switch to prepare for the change of output tapes for the sequence next to be formed (251). It reinitializes the stepdown counter and switch for the next series of stepdowns and sets sw 1 to N. The program branches through sw 1 (255) to compare the new record from the last stepdown tape with the others ranked for the new sequence. When its position in the ranking is determined in the usual fashion, the RDW of the first record of the new output sequence is found in BRECORD, and the processing of it continues in normal fashion from BRECEXIT.

OUTPUT END-OF-REEL ROUTINE

When a PUT goes to the file scheduler for a new output block and EOF is indicated from the previous write operation, the output end-of-reel routine is entered. Phase II bypasses the IOCS end-of-reel routine and branches to its own routine (286). The main functions of this routine are to CLOSE the current output file (292) and to find (286) and set up for (293) the next sequential and not EOF output tape as specified by the tape table. It also notifies the user, however, if there are too many records for the available tape and order of merge. If M output reels are full (288) the sort comes to unconditional HALT 2103. If M−1 output reels are full (290), the sort comes to HALT 2104, which warns the user of the risk that Phase II may not be completed successfully. If he did not encounter halts warning him of similar difficulties in Phase I, it may be that he has short reels of tape. If short output tapes are suspected, the user is advised to replace them and restart the current pass. If, instead, his choice is to proceed after HALT 2104, and if this pass is successfully completed, HALTS 2000 or 2001 may be encountered at the beginning of another Phase II pass. (See the next section.) Although output EOF may occur in the middle of a sequence as determined by the merge network, parts of it which are written on different tapes will be separate sequences; therefore, the sequence count is increased by one to count the portion just written (287).

Since neither the input nor output tapes of Phase II make normal use of the IOCS end-of-reel routine, if IOCSEOREX is ever used by any of them it is an error, and unconditional HALT 2101 follows (294).

Description of Chart 6: Beginning-of-Pass

In Phase II there are many steps which would normally be included in the assignment routines but which must be repeated at the start of every pass. These will be referred to as the beginning-of-pass procedures. Before each pass the user is allowed to exit at BSTARTPASS (216) for any added programming he may desire. Next, as the first step in the pass initialization, provision is made for the switching of input and output tapes (217). Controls are modified to allow the previous read tapes to become write tapes, and vice versa. This includes changing the channel and unit members in the input DTF's (218).

Then a test is performed to determine whether Phase II will continue with another pass or whether Phase III will be loaded (219). If the number of sequences is less than or equal to the order of merge, Phase III will be called (221). Note that the number of sequences from Phase I may be no greater than the order of merge; if so, Phase II processing will be completely bypassed. Before calling Phase III, however, the user is allowed to exit at BENDPHASE2 to enter any coding he may desire (220). If the number of sequences is greater than the order of merge, the program proceeds to calculate the additional Phase II passes required (222). It then checks to see whether the number of passes required has been reduced since the last pass (223). If not, it also checks to see whether the number of sequences has been reduced since the last pass (224). If so, the program comes to HALT 2001 (226), and, if not, it comes to HALT 2000 (225). After the latter halt, the user should not proceed, as he may be in an unending sort as discussed previously. After either type of halt, however, the user may elect to proceed simply by pushing START. If the user chooses to restart from the beginning of the last pass, he simply employs the normal procedures for using IOCS restart.

If the number of passes required has been reduced (223), or if not and the user decides to proceed anyway, the program continues to BSUMMTEST (227). At BSUMMTEST the user may exit to a decision routine which will decide whether to summarize this pass or not and will branch accordingly to BDOSUMM (229) or BSKIPSUMM (230). If the user does this after punching zeros in the summarizing-pass-control column of control card 2, he will come to HALT 2002 whenever BDOSUMM is selected. Such zeros indicate no summarizing in Phase II, and no compare routine (BCMPSEQW) for this purpose was generated during assignment. The

user may push START, however, and proceed without summarizing despite his mistake. If the user has chosen not to exit at BSUMMTEST (227) for pass-to-pass control of summarizing, the sort continues to a check of the control information just mentioned (228). If the number in the pass control field (APASSCNTRL) is not less than the number of Phase II passes remaining, the sort will branch to BDOSUMM; otherwise it will branch to BSKIPSUMM. This allows the user to summarize on the last *xx* passes or to skip summarizing entirely. The instruction BDOSUMM (229) sets BSUMMSW 3 (247) to Y, so that BRECEXIT (272) will be set (248) for summarizing after all the first GET's are complete.

Hash and record counts are saved for later comparison, and the counters are reset (230). Index words are reset to allow switches 1, 2, 3, and 4 to be set to N again (230). BRECEXIT (272) and BSTPDWNCHK (240) are reset for the first GET loops (231). Finally, the RDW's associated with each block must be generated or regenerated (232).

To prepare for writing the checkpoint record, the sort initializes the output DTF for the last output unit and channel and zeros are placed in the OPENPROC and LABELINF entries (233). The sort sets the checkpoint writing routine for the channel and unit of the last output tape (233-1) and spaces this tape forward one segment mark (233-2). If no segment mark is encountered on this reel, the sort comes to unconditional HALT 2105. When it finds a segment mark, the checkpoint record is written (233-2). Input files are opened (233-3). The sort opens the last output file (234) and tests whether all are OPEN (234-1). Since they are not, it sets the DTF for the next output tape (234-2), restores the OPENPROC and LABELINF in the DTF (234-2), and opens the next output file (234). This loop is repeated until the test (234-1) shows that all output files are OPEN. The DTF is then reset to the last output tape, and the creation date of the output tapes is saved for input label checking on the next pass (234-3). The merging for this pass is ready to begin.

Assignment Program, Chart 6

The assignment program of Phase II begins with a check of the Phasekey (200). Usually BPHASEKEY will contain a constant placed there during Phase I assignment. If so, the assignment program continues in normal fashion. However, if this is a restart after Phase II was interrupted on an earlier run, BPHASEKEY contains a constant inserted during Phase II assignment of the earlier run. On the restart run the restart tape contains all the information necessary for Phase II to be completed when the proper input reels are used. Phase II will be completed before any of the Sort 90 program is loaded; therefore, when at the end of Phase II the sort calls for Phase III, Phases I and II will have to be read and bypassed before Phase III is reached. If it is determined that it is a restart, the program branches to HALT 2003 (200-1). If alteration switch 1 is on (201), and START is pressed, the program continues Phase II assignment. If this switch is off, pressing START causes the program to branch to load Phase III.

Next, it is necessary to modify several instructions related to IOCS (202). The IOCS OPEN routine will be left in storage during the execution of the running program of Phase II, rather than being overlaid by the records read as in Phases I and III. This permits tapes to be reopened at the start of each pass. Therefore, the OPEN routine occupies a different area of storage than it did in Phase I. Since there are three instructions in the EOR routine that refer to locations in the OPEN routine, it is necessary to modify these instructions to reflect the new location.

SPOOL may be operating and may cause a priority interrupt when interrupt is again allowed near the end of Phase II assignment. Since the main parts of IOCS (including the channel scheduler) remain in storage between phases, the force-switch off exit is still set to branch to the location of the pending switch of a file scheduler in Phase I. But this scheduler no longer exists, and the branch to one for this phase is not yet created by an OPEN. Therefore, to allow interrupts by SPOOL before the OPEN's for the first pass are complete, the sort resets the force switch in each channel scheduler to branch to the bypass switch (202).

The assignment program then comes to HALT 2111 (not shown) to allow changing the last input reel if this reel is on a tape unit whose channel and unit number is the same as one of the work tapes specified and if there are enough sequences so that Phase II will be executed. At this point the user may exit from the assignment program and enter optional coding (203). This exit is called BASSIGN.

Other once-per-phase jobs include preparing for the taking of hash totals if these are used (204). It is necessary to specify which field is to be hashed and to modify the necessary instructions. The program tests to see if any SPOOL is to be run during this phase and if so, appropriate modifications are made to instructions. Many characteristics of Phase II depend upon the order of the merge; therefore, this assignment program determines the order of the merge and loads index words and other instructions to determine such things as the number of output tapes, to set up the number of compare routines to generate, and so on (205). The running program also varies according to the form of the records to be used. The biggest

variation occurs between form 3 and forms 1 and 2 (206). For form 1 or 2 records, the assignment program calculates the number of RDW's to be generated for each tape and a total number (207). With form 3 records, the assignment program places the record length field indicator in the appropriate GET and PUT instructions, since form 3 record length varies (208). It is up to the user to specify the collating sequence, so that the assignment program can modify the switches and the skeleton instructions for the comparison routines (209). Assignment varies according to whether the user has decided to summarize this phase or not. If APASSCNTRL is zero, the user has decided not to summarize during Phase II. Therefore, a switch is set which causes HALT 2002 to follow if BDOSUMM is entered. If APASSCNTRL is not zero, assignment sets up to generate the compare routine BCMPSEQW; this includes specifying the control data segments it will use for comparison.

One of the major jobs of Phase II assignment is to generate the compare routines and file schedulers and to modify output and input DTF's (210). The number of compare routines needed to determine the ranking depends upon the number of control data segments. Using model instructions and constants, the routines are assembled in a work area and moved into the appropriate positions in the running program. File schedulers provide the necessary linkage to make use of IOCS for input and output. Input file schedulers, as many as the order of merge, are generated in a fashion similar to that described for the compare routines. The skeleton of the first input scheduler lies in the position it is to occupy in the running program. Needed additional schedulers are moved to the desired locations and the appropriate modifications are made. The related DTF's are also created at the same time. Note that compare routines and file schedulers are located at the end of the running program. Space which is not actually used because the order of merge is low is used for records being sorted. When compare routines are short—because of few control data segments—even a bit more space is available following the last compare routine.

The next task of Phase II assignment is to initialize index words (211) for a variety of jobs such as the control of entry into the compare network in accordance with the order of the merge, end-of-file, stepdown, and first GET.

Then the sort saves the OPENPROC and LABELINF entries from the output DTF (211), since they will be altered by the restart assignment and checkpoint routines and are needed in their present form to OPEN all the output files except the one containing the restart routine. The sort sets the output DTF for the last output tape and writes the IOCS restart routine on it (211-1). The first output channel in Phase II is the A channel. The last tape on each channel now contains the restart routine, since Phase I assignment wrote it on the output channel of that phase, channel B. Now, no matter which channel is output during Phase III or any pass of Phase II, the restart routine will be available as the first record, after labels, on the last output reel.

The sort initializes the segment mark writing routine for the proper channel and unit and writes a segment mark on the checkpoint tape following the restart routine (211-2). The tape is then rewound to the load point. If an error occurs during the writing of the segment mark, the sort comes to HALT 2108. Pressing START causes the tape to backspace and then skip forward before returning for another attempt to write the segment mark. With success in writing the segment mark the sort branches to the beginning-of-pass initialization (216).

Exits and Modifications

As in Phase I, Sort 90 can be modified by changing constants and by adding new programming steps. Detailed instructions on modifications are given under "Modifications" in J28-6096, but main types of changes are listed here. If programming is added, exits at the following points are recommended:

1. During assignment the user may exit for any assignment functions to be added. The exit is BASSIGN (203).
2. From pass-to-pass, an exit is available for added beginning-of-pass initialization. This exit is BSTARTPASS (216).
3. From pass-to-pass, an exit is available to enter a decision routine to choose whether to summarize this pass. This exit is BSUMMTEST (227).
4. While records are being sorted, exits are available for processing of all records during a pass when no records are to be summarized, and for records to be summarized, during passes set for summarizing. These exits are BRECEXIT (272) and BSUMBRANCH (275).
5. Exit to added coding may be made at the end of Phase II before Phase III is loaded. This exit is BENDPHASE2 (220).

Only one change to constants is recommended for Phase II. When coding is added to the running program, the user should make a change to allow the additional storage areas to be written on tape by the checkpoint routine.

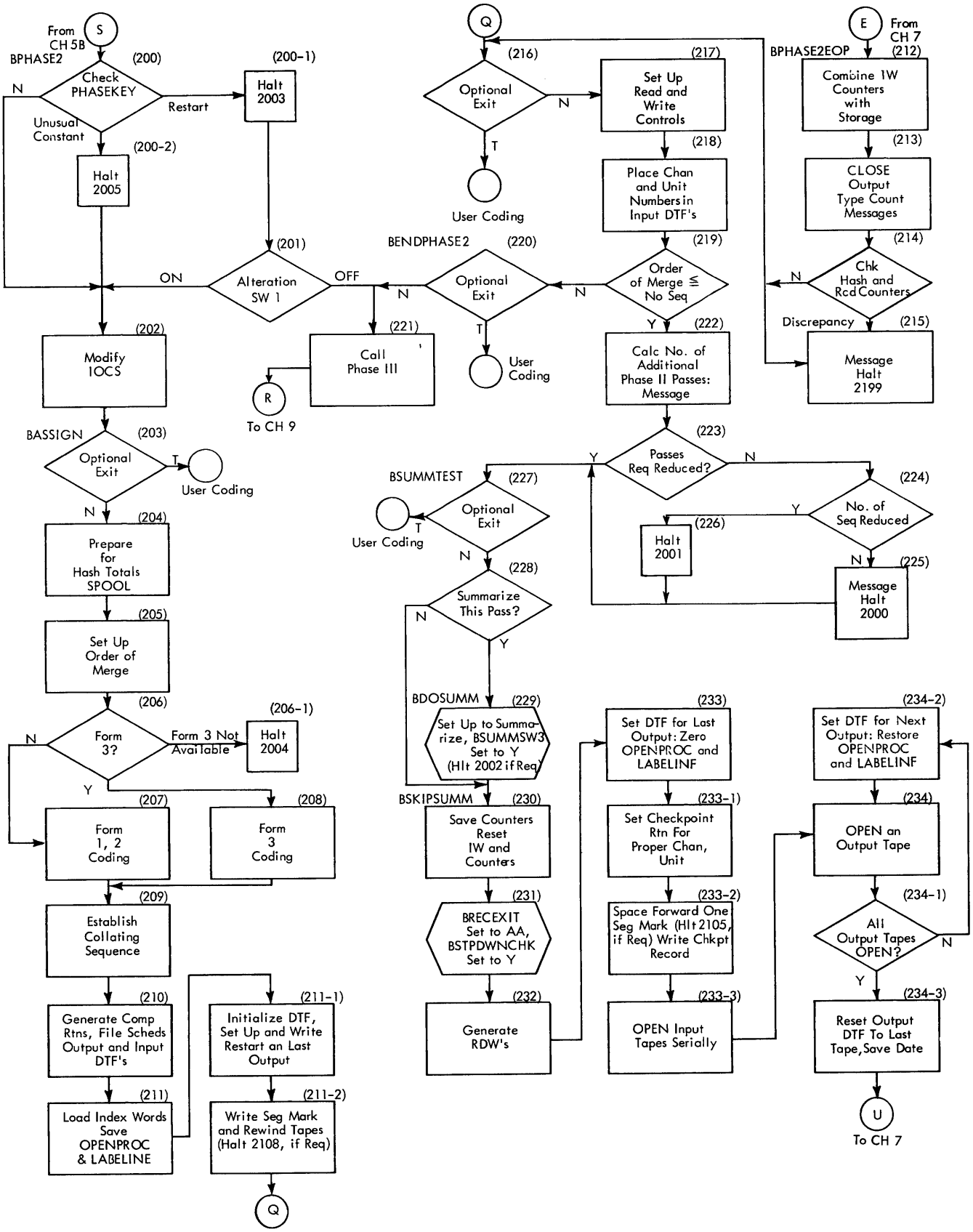
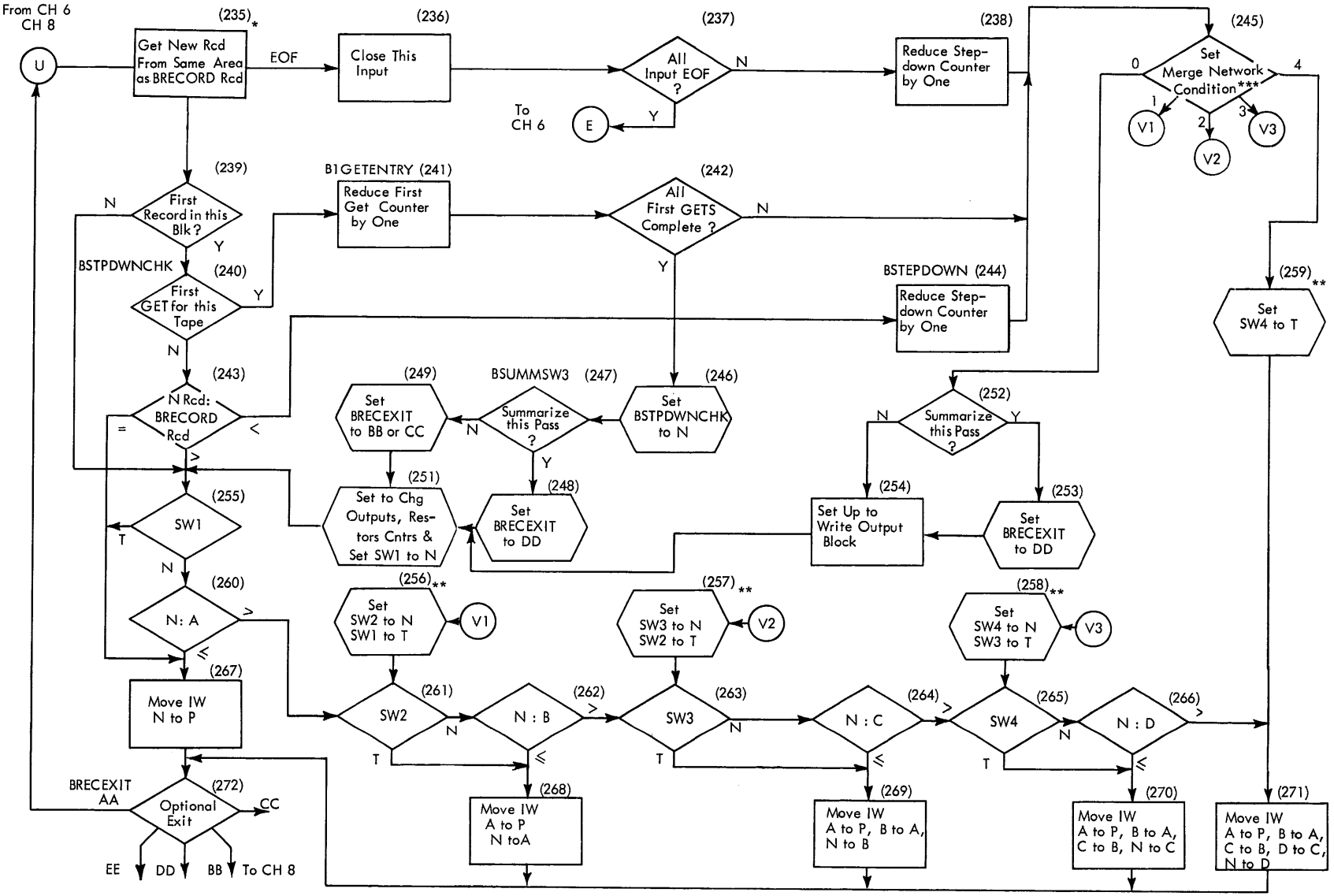


Chart 6. Phase II; Assignment, Beginning-of-Pass, and End-of-Pass Routines

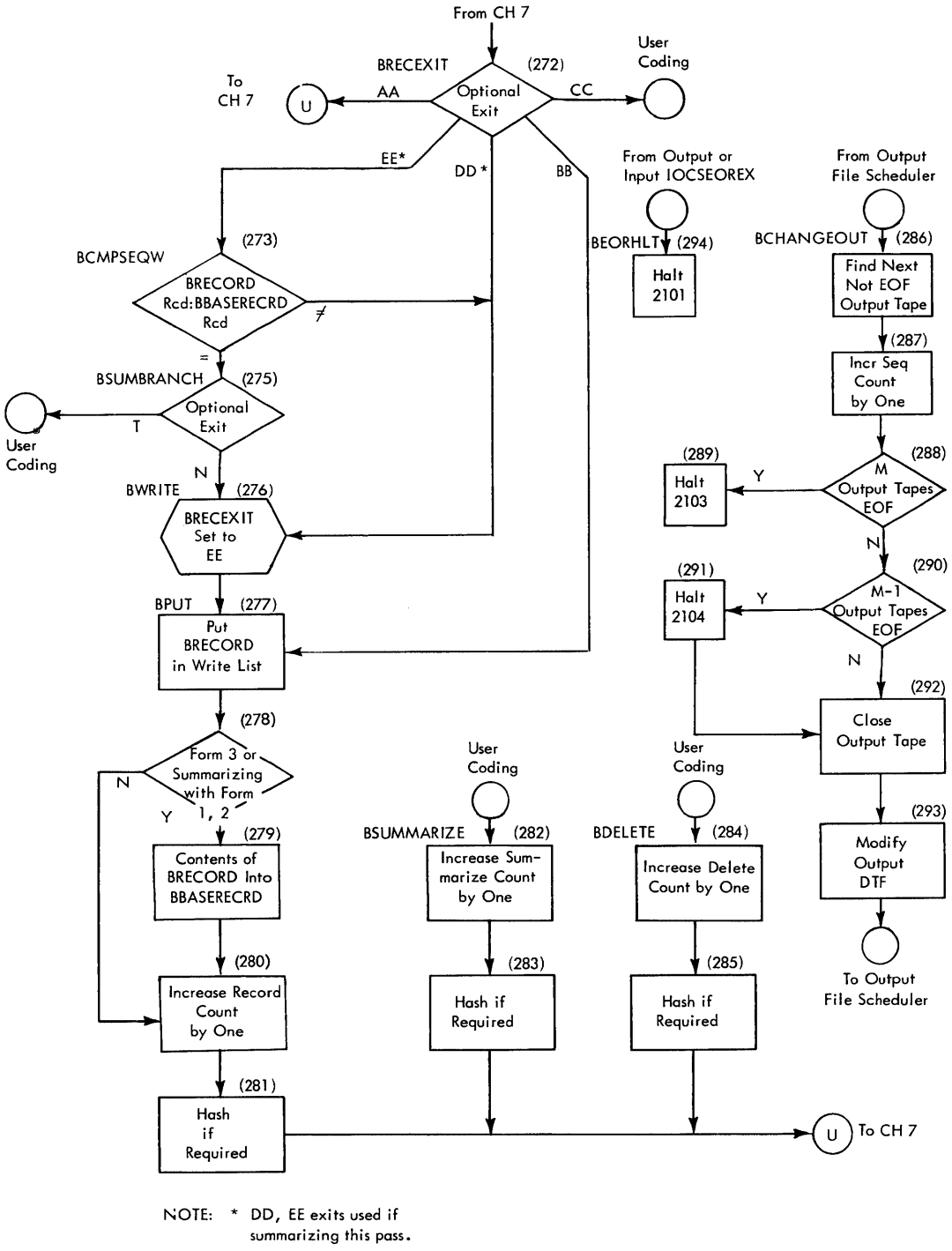


NOTES: * To fill the ranking of the beginning of a pass, GET a new record from each tape in turn and treat as a stepdown. BRECEXIT is set to AA.

**Once SW4, 3, 2, or 1 is set to "T" by an EOF, that switch will not be returned to N until the beginning-of-pass procedures of the next pass.

*** According to the number of tapes not first GET, not stepdown and not EOF

Chart 7. Phase II; Running Program, 1



NOTE: * DD, EE exits used if summarizing this pass.

Chart 8. Phase II; Running Program, 2

Phase III

Phase III of Sort 90 merges the output tapes of the final pass of Phase II into a single sequence and gives the user a final opportunity to add routines for summarizing, deleting, inserting, and editing records. (It is possible that Phase II is bypassed; if so, the input tapes to Phase III are the output tapes of Phase I.) As for Phases I and II, there is both a running program and an assignment program, the latter being executed first and overlaid with RDW's and records during the running program. Phase III is quite similar in its operation to Phase II, the principal differences being as follows:

1. Phase III is simpler, since each input tape contains a single sequence, and no provision for handling stepdowns is required, except as a sequence check.
2. It takes a single pass; therefore it is not necessary to retain the OPEN subroutine during the running program nor is it necessary to divide the initialization into an assignment section and a beginning-of-pass section.
3. Phase III allows for the insertion of records and for editing which will change the length of the record.
4. With the OPEN removed, there is more storage available for any added coding in Phase III.
5. The blocking factor for output is set to that specified by the user, and the use or non-use of output labels is at the option of the user.

Illustrations in this section assume an ascending sequence.

Running Program, Charts 10 and 11

For forms 1 and 2 the techniques of RDW interchange and the ranking of current records from each tape are employed in Phase III in precisely the same way as in Phase II. The symbols for the index words of the ranking are the same as for Phase II, except that they have the first letter "C" rather than "B." (See Figure 8.)

FIRST GET LOOPS

Following assignment, the program branches to GET a first record (321) from the first input tape. Since the block is empty, the program branches to the routine which fills a new block. When this is full, an RDW for the first record is placed in CNEWRECORD, and CBRNCHGETN is loaded with a branch instruction to the GET for this file. sw 1 (329) has been initialized to transfer to an entry routine (330), which is set at first to branch through line M−1, where M is the order of merge. The branch is to one of the switches sw 1, sw 2, sw 3, or sw 4; this switch is at T as initialized; and the switches numbered lower are at N as coded. Therefore, the program goes directly to one of the move routines and, after moving the irrelevant contents down the ranking one pair of words, unloads the contents of CNEWRECORD and CBRNCHGETN into the pair of positions at the top. For example, if the order of merge is three, locations 334, 335, 340, 341, 342, 346, 347 do not exist, and the entry at 330 is along line 2 to sw 3 (339) and the related move instructions (345).

The program proceeds to CRECEXIT (348) where it branches to GET (321) a first record from the next tape block. After the records are read into the appropriate block and the RDW and branch for the first record in this block are obtained, the sort again enters (330) the compare network, but through line M−2. This time the switch is set to allow this new record to be compared and then ranked with the first. If the order of merge is three, the flow is to 337, 338, and either 344 or 339 and 345—the latter only if the control data of the new record are larger than those of the earlier record. At CRECEXIT (348) there is a branch to another GET (321).

FIRST RECORD PUT LOOP

When a first GET is performed on the last input tape, the entry is through line 0. sw 1 is set to N (331), and CRECEXIT (348) is either set to CC or, if there is added coding, to BB. Then the new record is compared with the record whose RDW and branch are in pair A. Following any further comparisons which may prove necessary, the RDW and branch for the new record are inserted in their appropriate positions in the ranking, and the first record of the output sequence is represented in the lowest positions of the ranking, CRECORD and CNEXTGET.

After any coding added from CRECEXIT, the sort is ready for CSEQCHECK (349). Since this is the first record to be PUT, the program bypasses the sequence check and goes directly to any coding that has been added at CHIBRANCH (353). The sequence check (349), however, is now set to operate with all remaining records (354), and the RDW of the first record is entered

in the output list for form 1 and 2 records (355) or the record is PUT in the output block for form 3. CRECORD is entered into CBASERECRD (355) to provide the basis of comparison for the next sequence check, and the record count (356) and hash total (357) are appropriately increased before the program branches to GET (321) another record from the same tape as the record just PUT. This is the start of a normal loop.

NORMAL LOOP

In the normal loop of the merge a new record is obtained (321) from the same area as the record just PUT. The sort enters the compare network from the N branch of sw 1 (329), and the new record is ranked in the usual way. After the index words are moved, the remainder of the loop is like that of the first record PUT, except that the sequence check (349) functions normally. The comparison is not bypassed. The user may exit from CSTEPDOWN (350), CSUMBRANCH (352) or CHIBRANCH (353) to add coding which may vary according to whether the merged record (CRECORD record) is less than, equal to, or greater than the previous one (CBASERECRD record). If it is less, a record is out of sequence, and the user may want to add a special error routine to take account of this fact; any such programming should return the user to CWRITE (354). If the user chooses not to exit here, he receives a message indicating a stepdown, and the sort comes to HALT 3109 (351). Leaving alteration switch 1 off (351-1) and pushing START, the user causes the sort to type a message, "SEQUENCE BREAK TO OUTPUT" (351-4), and to branch to CWRITE (354). The out-of-sequence record will be PUT in the output file. If alteration switch 1 is on (351-1), pushing START causes the sort to type out both the stepdown record and the base record against which it was compared before coming to HALT 3110 (351-2). When alteration switch 1 remains on (351-3), the next START results in a message that the stepdown record is dropped (351-5), and the sort bypasses the PUT routine by going through CDELETE (358). If alteration switch 1 is off, however, the sort types a message, "SEQUENCE BREAK TO OUTPUT" (351-4), and branches to CWRITE (354).

If the merged record is equal, he may wish to summarize or delete, while if it is high, he may wish to do another type of editing. For the former, provision is made for the user to reenter at CDELETE or CSUMMARIZE so that the merged record is not PUT and record and hash counts (358, 359, 360, 361) will be handled correctly without any special programming by the user. Otherwise reentry is at CWRITE (354). Should the user decide to insert records at any point by added coding, a separate subroutine is provided which PUTS the record to be inserted and also allows the proper handling of these counts (362, 363, 364, 365). The program branches to the insert subroutine and, when this is complete, return is made to the user's coding.

INPUT END-OF-FILE

When the next GET is attempted (321) and an EOF signal is encountered, the program branches to CLOSE that input file (322). A test (323) is made to determine whether this is the last tape to become EOF. If not, the program continues to an entry (328) to the compare network which is determined by the *number* of tapes which have not yet reached the EOF condition. Upon entry, the switch in the compare network of this *same* number is set to T. This, together with moving the appropriate index words, reduces the ranking to handle one less current record. For example, if four tapes have not yet reached EOF, sw 4 is set to T (335), and all pairs in the ranking are moved down one pair of positions (347). Since the set of move instructions is used for other purposes as well, the contents of CNEWRECORD and CBRNCHGETN, which are not new, are unloaded into CRECORD and CBRNCHGETD. However, the setting of sw 4 to T prevents the comparison of new records with the record represented here and the execution of any further move instructions affecting these locations for the remainder of Phase III.

After the appropriate set of move instructions has been executed in an EOF loop, the next record to be PUT is represented in the lowest pair of the ranking, and the remainder of the loop from CRECEXIT (348) continues the same sequence of instructions as in a normal loop.

To perform the input end-of-reel processing just described, the sort leaves the IOCS end-of-reel routine through IOCSEOFEX and does not return. Therefore, any use of the IOCSEOREX for input end-of-reel is an error and leads to HALT 3101 (373).

OUTPUT END-OF-REEL

When a PUT goes to the file scheduler for a new output block and EOF is indicated from the previous write operation, the IOCS end-of-reel routine is entered. From IOCSEX1 the sort adds a routine which combines the index-word record counters with the output reel-by-reel storage counters, adds the record and hash totals to cumulative counters, and resets the reel-by-reel counters (371). The reel counts are entered in the trailer label areas, if output labels are used and require them. The user is then given the option of an exit at CENDEXIT1 (372), so that he may add further processing of the standard trailer label. After added routines or if no exit is taken, the sort returns to the IOCS end-of-reel routine.

Exiting from IOCSEOREX in output end-of-reel processing, the sort types an end-of-reel message (366). If all the M−1 output reels are full (367), HALT 3115 follows (368). This allows the user to change tapes. The sort is reset to write on the first output unit (368), and it writes successively on each of the remaining units, if necessary. The IOCS reel-change halt is disabled, the DTF is reset for the next tape unit, and the file serial count is increased (369). The user may add further coding from CEOREXIT (370) before return is made to the IOCS end-of-reel routine.

END-OF-PHASE ROUTINE

When the last input tape is EOF, it is closed (322), and the program branches (323) to CWINDUP (323-1) and the end-of-phase routine. The user may exit to added end-of-phase coding but must return to CLOSE the output file (324). The various record and hash counters which get separate totals for summarized records, deleted records, and inserted records are suitably combined (325). Hash and record count totals are checked against the totals of the last pass of Phase II (326). If discrepancies exist, a message indicates this, and the program comes to HALT 3199 (326-1). With alteration switch 1 off, the user may press START to restart Phase III. If the user chooses to set alteration switch 1 on and press START, the program will ignore the discrepancy and proceed to CENDPHASE3 (327) to conclude the sort. At CENDPHASE3 the user may exit to load or execute other programs. Alternately, the sort will continue to HALT 3333 (327-1), from which it will enter a busy loop (if START is pressed) to allow continuation of SPOOL.

Assignment Program, Chart 9

Phase III begins with a check of the Phasekey (300) and proceeds if CPHASEKEY contains the constant inserted during Phase II assignment. Otherwise, the program comes to HALT 3005 (300-1). If the user elects to proceed after such a halt, or if there has been no halt, a constant indicating Phase III is placed in CPHASEKEY. Next, if the user has so requested on the control card, and if Phase I produced no more sequences than the order of merge so that Phase II was omitted, the program will come to HALT 3111 (301) to allow the user to change the last Phase I input reel. The user may wish to save this file and still use this tape unit in further processing.

In Phase III, the OPEN subroutine will not be retained as it was in Phase II, but will be overlaid when records are read in for sorting as in Phase I. Its location in storage changes from phase to phase, and branch instructions in the EOR routine addressing it must therefore be modified (301). SPOOL may be operating and may cause a priority interrupt when interrupt is again allowed just before the checkpoint record is written (314-3). Since the main parts of IOCS, including the channel scheduler, remain in storage between phases, the force-switch off exit will be set to branch to the pending switch of a file scheduler in Phase II. But this linkage has not yet been created by an OPEN. Therefore, to allow interrupt by SPOOL before the OPEN's of assignment are complete, the force switches in the channel schedulers are reinitialized (301) to branch to the bypass switch. The user is then given an opportunity to exit at CASSIGN (302) for additions to the assignment program.

Next, the necessary information is obtained from the communications block to prepare the output header labels (303). This includes the file serial number, creation date and retention cycle. The contents of the hash and record counters are saved for later comparison and the counters are reset to zero (304). The sort checks as to whether or not hash totals are to be taken and if so, modifies the hash instructions according to the hash field specified by the user (305). If SPOOL is not to be run, assignment eliminates the SPOOL test in the channel schedulers.

The order of merge is obtained, and various instructions and index words which depend upon it both in the remainder of the assignment routine and in the running program are adjusted appropriately (306). For example, the program loads an index word controlling the number of compare routines and input file schedulers to be generated. It adds the order of merge to another index word which later in assignment will load other index words which act as counters. The entry to the compare network after the first GET is controlled in this way.

For form 1 and 2 records, assignment determines the number of RDW's per block and prepares for operations which depend upon this number, such as making the sign of the last RDW minus for tape operations (307, 308). Adjustments are also made for any change in record length by Phase III editing. For form 3 records (307, 309) the number of RDW's required is determined, and instructions which depend upon the maximum record length are set up. Counters are modified to handle form 3 records, and the length of the area to be provided for input file schedulers is increased by eight.

Instructions in assignment which are used to generate compare routines are modified in accordance with the collating sequence (310). The generation instructions are also modified according to whether one or

more than one control data segment is to be compared. The last output tape is eliminated as a sort output tape and is retained for use only as a checkpoint tape (311). Thus, there are only M−1 output units for Phase III. Assignment assembles the output DTF with the proper channel and first output unit, tape density, block size, record length, unreadable record procedure, and the other necessary information (311). The sort generates the required compare routines, input file schedulers, and input DTF's (312). Major characteristics of this generation are the same as in Phase II assignment.

Next, the program loads a number of index words (313). For example, it initializes sw 1 to the OPEN entry to be used following each first GET. The sequence check routine is set to bypass the comparison until after the first record is PUT, and the switch restoring it (354) is readied. The instructions (332, 333 and 334) used to set sw 1 and sw 2 during an EOF entry are initialized, and counters for the first GET and EOF entries are set according to the order of merge.

Sort 90 assignment prepares for RDW generation by finding the starting point for the RDW lists and the number of input and output RDW's (314). It loads this information into the appropriate instructions. Note that output blocking and record length may be different from that found during sorting. In preparation for writing the checkpoint record, the checkpoint tape spaces forward one segment mark (314-1) and the checkpoint writing routine is set for the proper channel and unit (314-2). If no segment mark is found on the forward space, the program comes to unconditional HALT 3105. Normally, however, the segment mark forward space allows the sort to pass over the restart record and write the checkpoint record as the next record (after the segment mark) on the checkpoint tape (314-3). Assignment restores the capacity of IOCS to write output header labels (314-4). If form 3 records are being processed (315), the program generates 2(M+1) RDW's (316); each RDW represents the maximum block size when processing form 3 records.

The OPEN routine is modified to prevent the signs of RDW's from being set, since at present this would modify locations in the assignment program (316-1). The first output and all input files are opened (317, 318), and the sign setting function is restored to the OPEN routine (318-1), before the user is able to exit at CADDASSIGN (318-2). For form 1 and 2 records the RDW's are generated (319, 320); the RDW's are created in the index word CBUILDRDW, the proper sign is added, and the index word is unloaded into the proper location.

Exits and Modifications

Phase III may be modified by changing constants and adding new programming steps. Detailed instructions on modifications are given in J28-6096, under "Modifications." As in Phase I, the necessary DTF's, DC's, DA's, file schedulers, IOCS macro-instructions, and required linkages may be added to allow the reading or writing of additional tape files.

The following exits may be overlaid by branches to added routines:

1. At the beginning and end of the assignment routine, exits are available for such functions as opening additional tape files. These exits are CASSIGN (302) and CADDASSIGN (318-2).
2. During sorting, extensive provision is made for summarizing, deleting, and editing of records as well as special treatment of (unexpected) stepdown records. The recommended exits are CRECEXIT (348), CHIBRANCH (353), CSUMBRANCH (352), and CSTEPDOWN (350).
3. During the output end-of-reel routine the user may cause an exit to his own routines for additional trailer label processing. The exits are CENDEXIT1 (372) and CEOREXIT (370). These replace the IOCSEOREX and IOCSEX1 used by Sort 90. The other IOCS output end-of-reel exits are available to be used in the normal way, if desired.
4. Two exits to added end-of-phase programming are available; further end-of-job routines or linkage to other programs may be entered. The exits are CWINDUP (323-1) and CENDPHASE3 (327).

During the merge of Phase III, added programming may call for the insertion of records into the sequenced output file. An insertion routine beginning with CINSERT (362), PUTS the inserted record and keeps separate counts of records and hash totals.

In Phase III, constants may be modified to modify the IOCS handling of output files and tape labels. The user may control whether the tape is rewound and unloaded or not and may change the way the sort writes trailer labels.

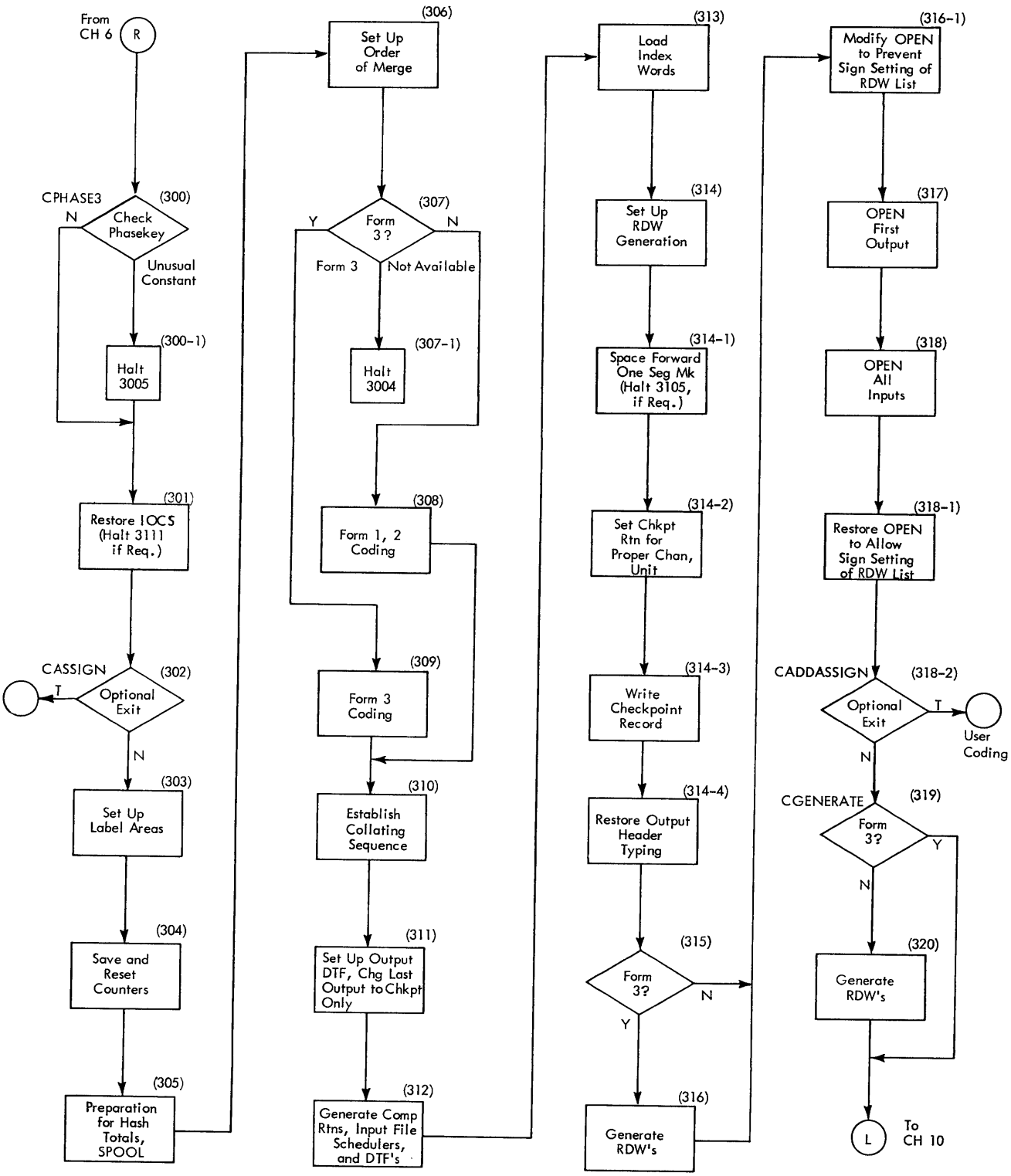


Chart 9. Phase III; Assignment Routine

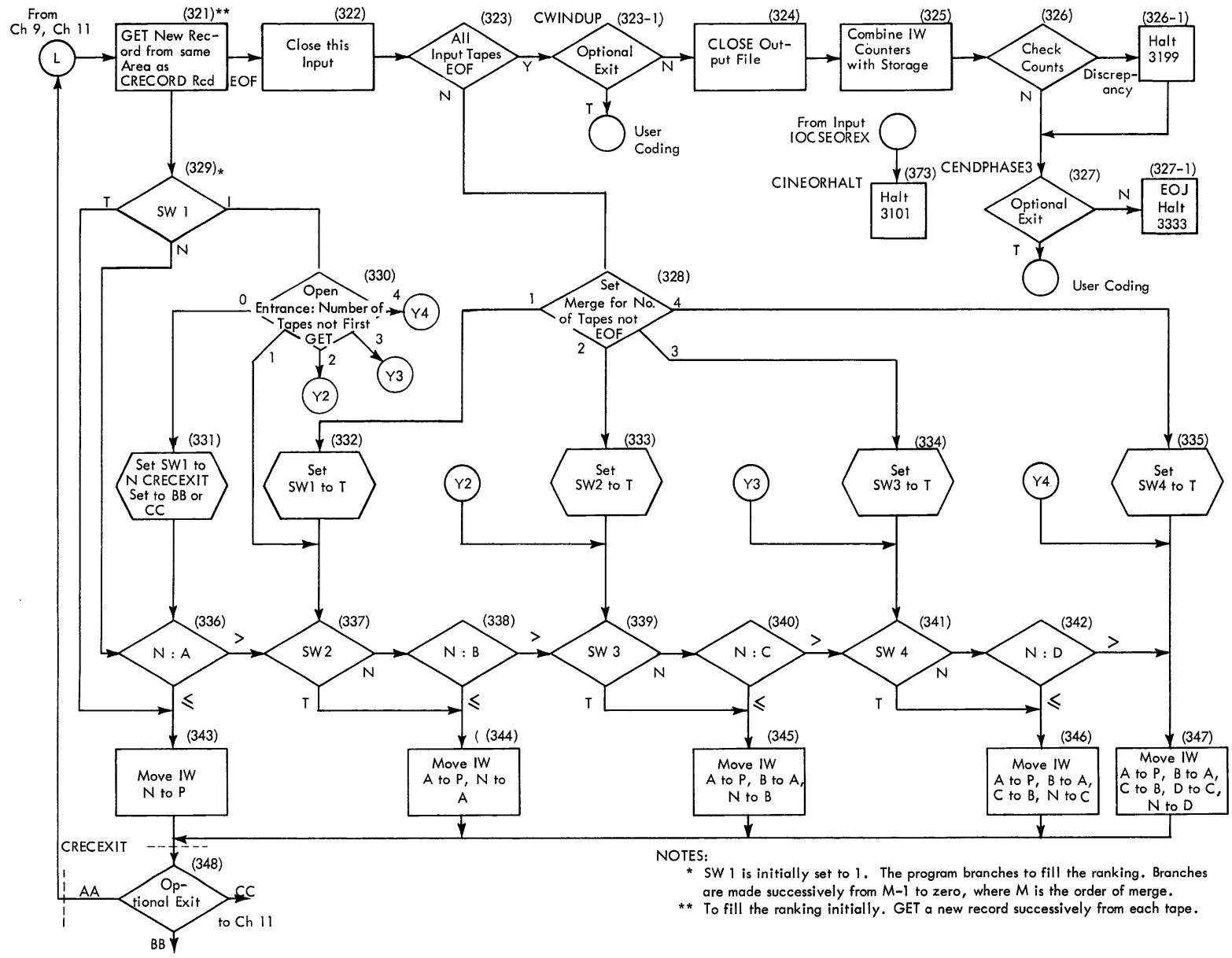
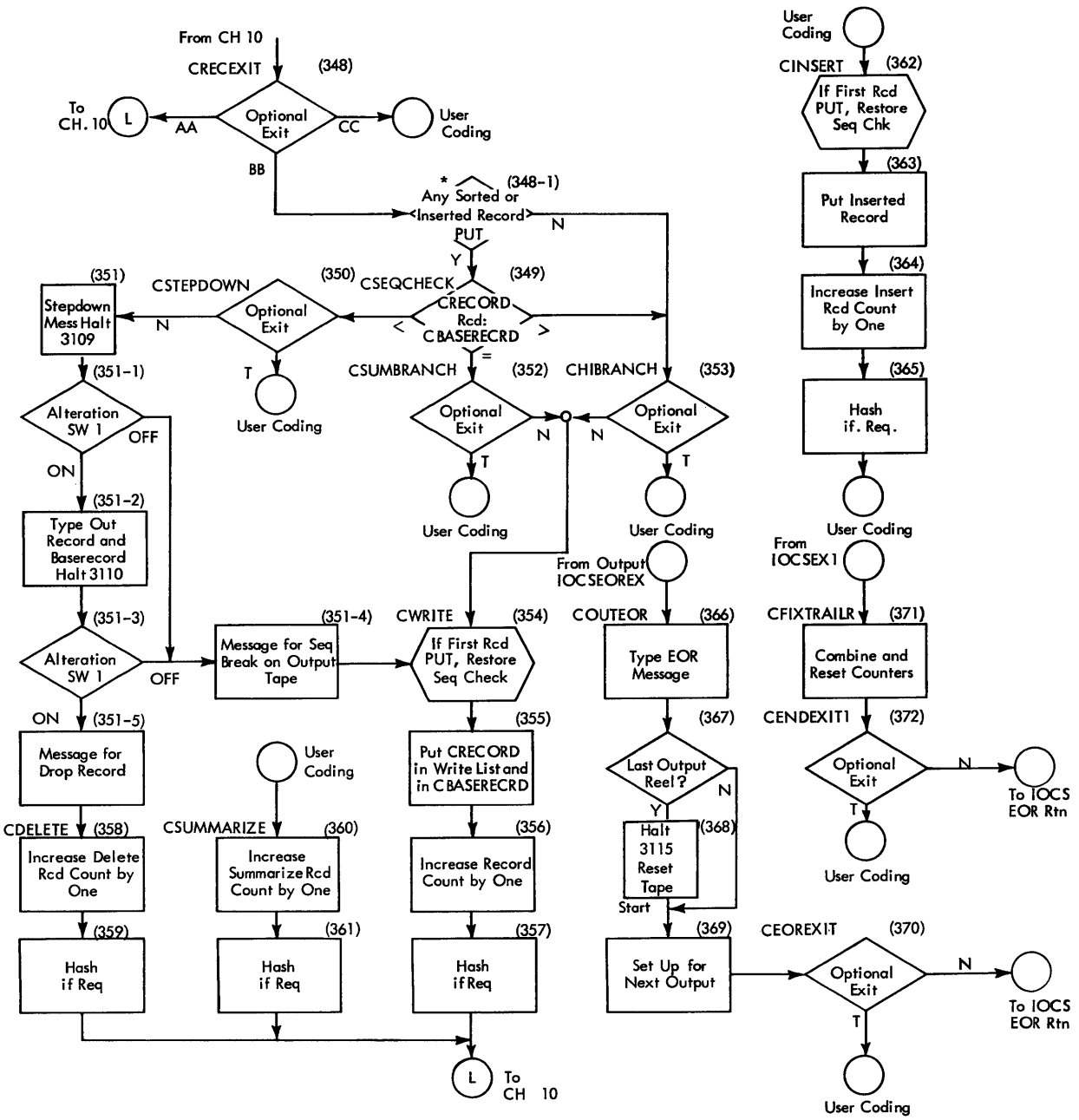


Chart 10. Phase III; Running Program, 1



NOTES:

* This switch is actually included in the compare routine which begins with CSEQCHECK, but it causes the comparison portion to be bypassed by a branch to CHIBRANCH until after a record has been PUT.

Chart 11. Phase III; Running Program, 2

Checkpoint and restart procedures may aid the user in several ways. Machine malfunction may result in a hang-up or the kinds of conditions which result in a specified sort halt. If the malfunction was transient (or if not, when it is corrected), the user may be able to restart from a point prior to the error condition, but after the beginning, and proceed successfully. It is not necessary to repeat the earlier stages of the sort which were completed correctly. Checkpoint and restart also allow the partitioning of a long sorting job, so that successive stages may be run at different times.

Sort 90 makes two kinds of restart provisions:

1. Phase 1 provides its own reel-by-reel restart procedures for labeled input records. If it finds discrepancies between the trailer label and internal counts for a completed input reel, halts follow. As one option the user may choose to restart simply by setting alteration switch 1 properly and pushing START. The sort rewinds the input reel, repositions the output tapes, and repeats the processing of this reel. Restarting for other reasons in Phase I requires that the sort be started again from the beginning.
2. In Phases II and III, Sort 90 uses the IOCS checkpoint and restart routine. The following paragraphs describe its application to Sort 90.

Sort 90 writes a checkpoint record only when it gives a checkpoint macro-instruction. It writes a checkpoint record at the beginning of each pass of Phase II and during assignment in Phase III. Restart of a given pass in Phase II is possible only as long as the records on the input tapes remain as they were at the beginning of that pass. Since writing the output of one pass destroys the records on the input tapes of the previous pass, restart is possible only from the beginning of the current pass. For this reason also, Sort 90 retains only one checkpoint record on a given checkpoint tape and writes the new checkpoint record over any previous one on this tape.

Sort 90 takes the option in Phase II of writing checkpoint records on an output tape which will also contain sorted records. Phase III uses one of the tapes on the output channel but limits its use to checkpoint only. Since input units and output tapes alternate from pass to pass in Phase II, the sort must have two tapes for checkpoint records, one on each channel. In Phases II and III the unit chosen for checkpoint is the last one on the output channel. The last tape for a given channel is the one entered last on the control card.

Sort 90 does not use the normal procedure for loading and readying the checkpoint and restart routine. Normally the assignment section, the checkpoint writing routine, and the restart program enter storage during program loading, but before the IOCS OPEN and IGEN routines are loaded. As soon as these parts of checkpoint are loaded, the assignment section prepares the checkpoint and restart routines in accordance with a DCHPT entry, and writes the restart program as the first record on the checkpoint tape. Then loading resumes, and the OPEN and IGEN routines overlay the locations just occupied by the assignment and restart sections.

A generalized program such as Sort 90, however, cannot fully specify the DCHPT entry until the control cards are read. Therefore, it must load the assignment and restart sections and retain them in memory until part of the sort assignment program (including DCHPT specification) is executed. Then restart is assigned and written as the first record on the checkpoint tape, after labels or segment mark. During processing, records overlay both the assignment and restart sections. The checkpoint writing routine, which is retained in storage during processing, is assigned at the same time as the restart program.

A checkpoint tape must be on both channels, but during Phase II assignment the restart program can only be written on a tape of the channel to be used for output for the first pass. This is channel A. Therefore, the restart for channel B, output channel for Phase I, is written during Phase I assignment. Phase I must load and assign the checkpoint and restart routine, although it writes no checkpoint records and cannot use the restart program.

The presence of the restart and checkpoint records on tapes containing records causes no difficulty in reading or writing the records to be sorted. If the input to Phase I contains checkpoint records, the user must punch "1" in column 65 of the first control card. In Phases II and III one of the input tapes always contains a restart record and, after pass 1, a checkpoint record. These records are read as long length records (LLR) and are bypassed accordingly when records to be sorted are read from this tape. Phase I opens output tapes during assignment after the restart routine is written; Phase II opens them after the checkpoint

record is written at the beginning of each pass. The sort does not write over checkpoint and restart records on the output tape because the tape is not rewound after the new checkpoint is written, and label processing is bypassed when this tape is opened.

Whenever the user decides to restart Phase III or a pass of Phase II, he simply follows normal IOCS restart procedure. He must use the restart initiator program, which will load the restart program from the checkpoint tape and transfer control to this program. The restart program positions tapes and restores storage; Sort 90 continues from the point at which the checkpoint record was taken.

The storage locations normally contained in the sort checkpoint record include the index words used by the sort and from BPHASEKEY or CPHASEKEY through the storage limit specified on the control card. To include programming modifications, the user must patch the RDW's following the DCHPT. Instructions for doing this are given under "Modifications" in J28-6096. A restart does not routinely restore all of storage because SPOOL may be operating.

Locating Errors in Sort 90

The sort is varied through control card punching and through more direct modification. The latter may involve added programming or simply overlaying constants. To avoid errors in running Sort 90, the user must avoid making errors in effecting these variations. If errors do occur, however, make the following checks:

Check 1

Check first for errors in control card punching. Although Phase I assignment of Sort 90 checks for many types of control card errors, it cannot check for all types. It will check for the following:

1. *Consistency among user specifications.*—For example, are the control data fields located so that they will lie within records of the length indicated?
2. *Conformance to rules.*—For instance, do the control field specifications indicate a total of not more than the maximum of 160 digits of control data?
3. *Support of the goals of sorting.*—Does the user indicate records or control data field of lengths greater than zero?

Sort 90 will *not* check the specifications punched in the control cards for the following:

1. *The configuration of components.*—Are the indicated tape units available? Are the storage limits indicated in the control card within the storage limits of the computer?
2. *The actual input.*—Are records of the form and length indicated? Do the control data lie where indicated?
3. *User's intended specifications.*—If the user wants the records in descending collating sequence, has he so indicated?

The unchecked errors may result in a variety of program malfunctions and computer hang-ups. On the other hand, the program may be executed with apparent success, with an error becoming evident only at some later time. Hence, despite the extensive checking which is programmed, care is required in the preparation for a new sort application.

Check 2

Check for misplacement of the control cards. Are they placed in the card reader in the proper order?

Check 3

If the user has modified the sort, check these modifications for programming errors. Check their placement in storage.

Errors in a New Application

If a hang-up is on the first run of the sort in a new application, suspect errors of the types already described, rather than machine errors. Always question whether any changes have been made from previous successful runs of the sort. If no changes have been made in the control card specifications or through user modifications, has there been any change in the characteristics of the input? A change in input may reveal errors not evident before, or it may require change in the control cards or change in any added programming.

Sometimes the hang-ups from such sources look very much like machine difficulties. For example, if the user has indicated that the work tapes will have labels, and in fact they do not, a read error will very likely result. When the program attempts to check the header label on the Phase I output tapes, it will not usually find a record of the proper length or density occupying the first block of tape.

When rerunning the sort to determine the source of errors, be sure to use a hash total check option which permits pass-to-pass checking. If labels permit, use hash total and record count checks on both input and output trailer labels, in addition.

Control Card Summary

Because errors in the preparation of control cards occur frequently, a summary of each control card is available for ready reference on the following pages.

Control Card 1

COLUMNS	NAME OF PARAMETER	CONTENTS	REMARKS
1	Input Channel	x	(x = 1 or 2). Input channel number.
2-6	Input Tape Units	$uu_1u_2u_3u_4$	u = input unit, u_1-u_4 are alternate units consecutively. Each unused alt. = 0. Zero unit cannot be last.
7	Input Tape Type	x	(x = 1, 2, 3 or 4). 1 = 729 II, low; 2 = 729 II, high; 3 = 729 IV, low; 4 = 729 IV, high.
8	Channel A	x	(x = 1 or 2). First work tape channel; should be same as column 1.
9-13	Channel A Tape Units	$uu_1u_2u_3u_4$	u-u_4 are work tape units on channel A. M* units must be specified. **Zero unit cannot be last.
14	Channel B	x	(x = 1 or 2). Work tape channel B number. Not same as channel A, unless one-channel system.
15-19	Channel B Tape Units	$uu_1u_2u_3u_4$	u-u_4 are work tape units on channel B. M* units must be specified. Zero unit cannot be last.
20	Output Tape Type	x	(x = 1, 2, 3 or 4). 1 = 729 II, low; 2 = 729 II, high; 3 = 729 IV, low; 4 = 729 IV, high.
21	Record Form	x	(x = 1, 2 or 3). Input record form.
22-24	Input Record Length	xxx	Number of words per record, form 1; or maximum number of words per record, form 2; or minimum number of words per record, form 3.
25-27	Input Blocking	xxx	Input blocking factor, forms 1 and 2; or maximum input block size, form 3.
28-30	Output Blocking	xxx	Output blocking factor, forms 1 and 2; or maximum output block size, form 3.
31	Zero Suppression Ind.	1	Output with zero suppression.
		0	No zero suppression.
32-33	Input Reel Count	00	No reel counts.
		xx	1-99 input reels. Must be specified for unlabeled file.
34	Record Count Ind.	0	No further action beyond pass-to-pass.
		1	Check input trailer record count.
		2	Insert record count in output trailer.
		3	Check and insert.
35	Hash Total Ind.	0	No hash totals.
		1	Check input trailer hash total.
		2	Make pass-to-pass hash total check.
		3	Insert hash total in output trailer.
		4	(1) and (2)
		5	(1) and (3)
		6	(2) and (3)
		7	(1), (2) and (3)
36-40	Hash Total Field	00000	No hash totals (if col. 35 = 0).
		wwwff	www = word number in record.*** ff = field control.
41-42	Unreadable Record Ind.	00	Halt for manual correction.
		xx	(xx = 10-29). Dump tape channel and unit number.
		50	Type the block before halt for manual correction.
		60	Retry under operator control.
43	Phase I Area Ind.	1	Phase I will decide.
		2	Two-area system.
		3	Three-area system.
44-47	Phase I Storage Limit	xxxx	4-digit core storage address.
48-51	Phase II Storage Limit	xxxx	4-digit core storage address.
52-55	Phase III Storage Limit	xxxx	4-digit core storage address.
56-60	Form 3 Length Field	00000	Form 1 and 2 records.
		wwwff	www = word number in record.*** ff = field control.
61	Input File Label Ind.	0	No labels.
		1	Low density labels.
		2	High density labels. If (1) or (2), control card 3 is necessary.
62	Work Tape Label Ind.	0	No labels.
		1	Low density labels.
		2	High density labels.

*M = order of merge for Phases II and III

**Channel A tape units may duplicate some or all input tape units (columns 2-6)

***First word in record is word 000.

Control Card 1 (Cont.)

COLUMNS	NAME OF PARAMETER	CONTENTS	REMARKS
63	Output File Label Ind.	0	No labels.
		1	Low density labels.
		2	High density labels.
			If (1) or (2), control card 4 is necessary.
64	Spool Indicator	0	No SPOOL.
		1	SPOOL possible.
			If (1), ES's 29 and 30 must be off.
65	Input Checkpoint Ind.	1	Checkpoint records in input file.
		0	No checkpoint records.
66-69	Phase I Edit Indicators	yxxx	(y = 1 or 2). Record form.
			xxx = record length for editing.
		0000	No Phase I editing or form 3 records apply.
70-73	Phase III Edit Indicators	yxxx	(y = 1 or 2). Record form.
			xxx = record length for editing.
		0000	No Phase III editing or form 3 records apply.
74-75	Reserved for User		Must be punched with zeros if not used.
76-80	Not Used	00000	Must be punched with zeros.

Control Card 2

COLUMNS	NAME OF PARAMETER	CONTENTS	REMARKS
1-6	Control Data Field 1 Position	wwwdll	www = word number in record.***
			d = starting digit in a word.
			ll = number of digits in the control field.
7-12	Control Data Field 2 Position	wwwdll	Same as above } Unused fields must be filled with zeros.
13-18	Control Data Field 3 Position	wwwdll	Same as above } Non-zero fields cannot be preceded by a
19-24	Control Data Field 4 Position	wwwdll	Same as above } zero field.
25-30	Control Data Field 5 Position	wwwdll	Same as above
31-36	Control Data Field 6 Position	wwwdll	Same as above
37-42	Control Data Field 7 Position	wwwdll	Same as above
43-48	Control Data Field 8 Position	wwwdll	Same as above
49-54	Control Data Field 9 Position	wwwdll	Same as above
55-60	Control Data Field 10 Position	wwwdll	Same as above
61-63	Total Control Data Digits	xxx	Sum of ll's in columns 1-60.
64	Collating Sequence	0	Algebraic, ascending.
		1	Algebraic, descending.
		2	Absolute, ascending.
		3	Absolute, descending.
65	Summarizing Equals Control	0	No cut-off for summarizing.
		x	(x = 1-9). Number of control fields to be considered.
66-67	Phase II Summarizing Pass Control	00	No Phase II summarizing.
		99	Summarizing every Phase II pass.
		xx	(xx = 01-98). Number of final Phase II passes for summarizing.
68-75	Reserved for User		Must be punched with zeros if not used.
76-80	Not Used	00000	Must be punched with zeros.

***First word in record is word 000.

Control Card 3, Input File Label

COLUMNS	NAME OF PARAMETER	CONTENTS	REMARKS
1-7	Reserved for User		Must be punched with zeros, if not used.
8-10	Reel Sequence Number	xxx	3-digit number* (1st reel of input file).
11-15	Creation Date	yyddd	yy = year file was created*.
			ddd = number of the day in the year.
16-20	File Serial Number	xxxxx	5-digit number*.
21-40	File Identification		Ten-character name in double-digit form*.
41-75	Reserved for User		Must be punched with zeros, if not used.
76-80	Not Used		Must be punched with zeros.

*Must be punched with zeros if no parameter.

Control Card 4, Output File Label

COLUMNS	NAME OF PARAMETER	CONTENTS	REMARKS
1-7	Reserved for User		Must be punched with zeros, if not used.
8-10	Retention Cycle	xxx	3-digit number*.
11-30	File Identification		Ten-character name in double-digit form*.
31-74	Comments		Punched in double-digit form*.
75-80	Not Used		Must be punched with zeros.

*Must be punched with zeros if no parameter.

Malfunctions Related to Input-Output

Machine Implications of Heavy I-O Usage

A sort program gives the tape I-O system a longer and more intense work-out than any other program likely to be in use at an installation, including customer engineer test programs. It is very common for a progressively deteriorating component finally to fail during a sort, even if it has previously evidenced itself in no other programs. In a typical sort, almost every type of tape command is issued many times.

Input-Output Control System (IOCS)

Errors may occur if the user attempts to make incorrect use of the IOCS routines which the sort contains or if he calls routines which are not available.

SECTIONS OF IOCS INCLUDED

The major parts of the IOCS which are included as an integral part of Sort 90 are:

1. CHAN2: The sort uses two channels and requires the necessary schedulers.
2. EOR1: The sort incorporates the EOR routine which permits the processing of tape labels.
3. CHPT: The checkpoint routine is used for restart in Phases II and III. (Phase I has its own reel-by-reel "restart" option.)
4. IGEN2: This routine permits SPOOL programs to operate with the sort, but the tape error routine to scan storage for illegal double-digit characters is *not* included.
5. OPEN2: The OPEN2 routine is used, but it is reloaded with each phase into different locations. In Phase II, OPEN is retained for pass-to-pass initialization, but in Phases I and II it is used only during assignment and overlaid during the running program.

These major parts of IOCS are available to the user if he should choose to read or write additional files during the course of the sort. However, as pointed out above, he must choose the right time to use the OPEN routine because sometimes it has been overlaid and other times it has been temporarily modified for use by the sort. All necessary information on when to exit is given the user. (See "Modifications" in J28-6096.)

The user should also note the absence of the IOCS double-digit scan. The IOCS error routine will not attempt to locate invalid alphameric characters, and the user should not expect to be able to call on the double-digit scan in conjunction with any added coding. Attention should be paid to the fact that write errors caused by invalid alphameric characters normally can be caused only by the user's editing modifications; the sort itself does not alter records.

KNOWLEDGE OF IOCS

Some understanding of IOCS is a prerequisite for understanding and using the sort. For example, knowing that a full block will not be written until the next PUT is initiated is helpful. It is the basis for understanding how summarizing into a record can occur even when the base record (into which information from succeeding records is summed) is the last record in a block. Although its RDW is in the output list which is ready for writing, the base record and other records controlled by this list will never be written until summarizing is complete.

Determining Machine Status When the Program Is Interrupted

This section can aid in determining the status of the computer and the program when the program is intentionally interrupted or when a hang-up occurs. Since it is possible for an interruption to occur during an input or output function, knowledge of IOCS switches, counters, and index words is also useful. (See *Program Systems Analysis Guide, 7070 Input-Output Control System,* C28-6119.)

The actual machine addresses assigned some of the symbolic locations discussed below are listed, for a particular assembly of the sort, in information provided the user; they are also given at the end of each phase in the program listing. The actual machine addresses of symbolic locations not thus given may be determined by consulting the listing in the section specified.

Phase and Pass

The user will know the phase and pass of the program from the typed messages received:

"7070 SORT 90" (003) indicates that the assignment program of Phase I is about to start.

"BxxxxGyyyy" (049) is given near the end of part 1 of Phase I assignment.

"Phase 1" (056) indicates that the Phase I running program is about to begin.

"Phase 2 PASSXX REQZZ" (222) indicates that the xxth Phase II pass is about to begin and zz Phase II passes (including the xxth pass) remain to be performed.

"Phase 3" (301) shows that Phase III is about to begin.

Other messages and any of the standard halts may also give the user aid in locating the stage to which the program has progressed. All halts and messages are listed under "Messages" and "Programmed Halts" in J28-6096, and many are shown on the flow charts.

The first location of the loaded program, which is usually 0325, is APHASEKEY, BPHASEKEY, or CPHASEKEY, depending on whether Phase I, II, or III is loaded. This location contains a constant indicating the phase presently loaded.

Channel, Unit, Block Count

The channel currently being used for input and that for output can be determined by reference to the proper DTF. The DTF's are located as follows:

PHASE I

Output DTF: IOCSFTBL02
Input DTF: IOCSFTBL01

PHASE II

Output DTF: IOCSFTBL01
Input DTF's: IOCSFTBL02, IOCSFTBL03, IOCSFTBL04, IOCSFTBL05, IOCSFTBL06

PHASE III

Output DTF: IOCSFTBL01
Input DTF's: IOCSFTBL02, IOCSFTBL03, IOCSFTBL04, IOCSFTBL05, IOCSFTBL06

In Phase I, the input DTF contains the channel and unit used for the file currently being read; the input tapes are read serially, and the DTF is reinitialized for the succeeding tape units. In Phase II and III, each input tape has its own DTF which gives the channel and unit information. In all phases, the output DTF contains the channel and unit used for the file currently being written. For each output unit in Phases I and II, words 1, 6, and 9 of a standard DTF are saved in a special area of storage. To switch from one unit to another the program inserts the proper set of words into the output DTF.

Phase III has only one output DTF, but reels are filled one at a time and the DTF is reinitialized for the succeeding tape units; there is no need to store words 1, 6, and 9 for reuse. In Phase II, channel and unit digits from the output DTF and input DTF's are exchanged, so the locations currently used for input and output DTF's remain unchanged from pass to pass. The DTF locations for Phase III also remain fixed and the channel and unit assignments are opposite to those of the last pass of Phase II.

The block count given in any DTF is the count for the unit concerned only. For Phases I and II, the same information for the inactive output units, which are not currently defined in the output DTF, may be obtained from the saved first words. The first and sixth words are stored beginning with the first output tape unit at ADTFINSERT for Phase I and BOUTPUT1 for Phase II. These locations are in the area used for constants unique to the phase being run; this area is just before the first instructions of the running program.

Record and Hash Total Counters

A number of locations are used in each phase for a variety of record counts. These counters are found in three different sections of the program for each phase. Some counters are also used to transmit information from one phase to another, so they are located in the common communications block at the beginning of the sort program in lower storage; this area is established in Phase I and remains in storage for all three phases. Other counters are found in the information areas unique to each phase; these areas precede the running program in each phase. A third set of counters are index words. The following three sections describe the functions of the counters, their location, and how counts may be taken if processing is interrupted.

Functions of Counters

Since tape labels are optional, as are hash total checks, some of the counts may be bypassed. The following list gives the function of the counters when they are used.

PHASE I

AWRCNTX(2,5) is reset to 9999 for each input reel, and it counts records by decrementing with a BDX instruction. When zero is reached, it is reset to 9999, and 9999 is added to AWRCNT. At input EOR the difference between its count and 9999 is also added to AWRCNT.

AWRCNT is reset to zero for every new input reel. It adds to its previous total 9999 when AWRCNTX(2,5) reaches zero and receives 9999—AWRCNT at input EOR time. Then, before being reset to zero, its contents are added to AWRITECNT.

AWRITECNT is set to zero during assignment, and it adds the contents of AWRCNT at input EOR time to its previous total. Its contents remain for Phase II where this count is moved to BPREVCOUNT.

ADLTCNT is reset to zero for every new input reel. It is incremented by one for every record deleted. At input EOR time its count is added to ADELETECNT.

ADELETECNT is set to zero during assignment and adds the contents of ADLTCNT to its previous total during the input EOR routine.

AOKHASH is reset to zero for every new input reel. The hash total for each record to be written is added to this location before the record enters the comparison routine of the scan.

AHASH is set to zero during assignment and adds the contents of AOKHASH to its previous total during the input EOR routine.

ADELHASH is reset to zero for every new input reel. The hash total for each record deleted is added to this location.

AINCOUNT is set to zero during assignment and adds the contents of ADLTCNT and AWRCNT to its previous total during the input EOR routine.

ADUMPCOUNT is set to zero during assignment and adds the contents of IOCSFTBL01(8, 9) to its previous total during the input EOR routine.

PHASE II

BWRITEX(2, 5), BDELETEX(2, 5), and BSUMMX(2, 5) are the record-by-record counters for records PUT, deleted, and summarized, respectively. These counters are set to 9999 at the beginning of each pass, and they count records by decrementing with a BDX command. When any one reaches zero, it is reset to 9999, and 9999 is added to BWRITECNT, BDELETECNT, or BSUMMCNT, respectively. These latter were set to zero at the beginning of each pass.

During the end-of-pass procedures the difference between each index word count and 9999 is also added to BWRITECNT, BDELETECNT, and BSUMMCNT, respectively.

BPREVCOUNT receives the contents of BWRITECNT during beginning-of-pass initialization. Before the first beginning-of-pass initialization, BWRITECNT contains the count of records written in Phase I as it was left in storage for transmission to Phase II. Thus, BPREVCOUNT saves the count of records written during the last phase or pass for comparison with the BWRITECNT of the next pass.

BHASHWRITE, BHASHSUMM, and BHASHDELET receive the hash totals on a record-by-record basis for records PUT, summarized, and deleted, respectively. They are reset to zero during beginning-of-pass initialization.

BPREVHASH is reset to the contents of BHASHWRITE during the beginning-of-pass initialization.

BCURRHASH is reset to hold the sum of BHASHWRITE, BHASHSUMM, and BHASHDELET at the end-of-pass, so that a comparison may be made with BPREVHASH.

BDUMPBLOCK receives during the input EOR routine the counts of the number of blocks dumped for errors. The counts added come from the DTF's of the completed reels.

PHASE III

CWRITEX(2, 5), CINSRTCNTX(2, 5), CSUMMX(2, 5), and CDELETEX(2, 5) are the record-by-record counters for records sorted and PUT, inserted and PUT, summarized, and deleted, respectively. These counters are set to 9999 during assignment, and they count records by decrementing with a BDX command. When any one reaches zero, it is reset to 9999, and 9999 is added to CWRITEREEL, CINSRTREEL, CSUMMCNT, or CDELETECNT, respectively; these latter are set to zero during assignment.

If trailer labels with record counts are to be written, the difference between each index word count and 9999 is also added to CWRITEREEL, CINSRTREEL, CSUMMCNT, and CDELETECNT, respectively, during the output EOR routine. CWRITEREEL is added to CWRITECNT and CEDITREC, and CINSRTREEL is added to CINSRTCNT and CEDITREC. CEDITREC is for the output trailer label. CWRITEX and CINSRTCNTX are reset to 9999, and CWRITEREEL is reset to zero.

Whether trailer labels are used or not, the same steps are carried out during the end-of-pass procedures.

Before it is set to zero in assignment, CWRITECNT contains the count of records written in the last pass of Phase II. This count is saved in CPREVCOUNT for comparison with CWRITECNT plus CSUMMCNT plus CDELETECNT at the end of Phase III.

CHASHREEL, CHASHINSRT, and CCURRHASH receive the hash totals on a record-by-record basis for records sorted and PUT, inserted and PUT, and summarized or deleted, respectively. At output EOR time (if labels are written with hash totals) and at end-of-phase time, in any event, CHASHREEL is added to CCURRHASH. At the same time, CHASHREEL and CHASHINSTR are combined into CEDITHASH for use in the output trailer label. The hash total from Phase II is found in CWRITEHASH and is transferred to CPREVHASH during the assignment. CCURRHASH is compared with CPREVHASH at the end of Phase III.

CDUMPBLOCK receives during the input EOR routines the counts of the number of blocks dumped for errors. The counts added to this cumulative total come from the DTF's of the completed reels.

Locations of Counters

Phase I Index Words	*Phase II Index Words*	*Phase III Index Words*
AWRCNTX	BWRITEX	CWRITEX
	BDELETEX	CINSRTCNTX
	BSUMMX	CSUMMX
		CDELETEX

Communications Block	*Communications Block*	*Communications Block*
AHASH	BHASHWRITE	CHASHWRITE
AWRITECNT	BWRITECNT	CWRITECNT
Information	*Information*	*Information*
ADLTCNT	BHASHSUMM	CEDITREC
AWRCNT	BHASHDELET	CEDITHASH
AOKHASH	BPREVCOUNT	CWRITEREEL
ADELHASH	BSUMMCNT	CHASHREEL
ADUMPCOUNT	BDELETECNT	CINSRTREEL
ADELETECNT	BDUMPBLOCK	CHASHINSRT
AINCOUNT	BCURRHASH	CPREVCOUNT
		CINSERTCNT
		CSUMMCNT
		CDELETECNT
		CDUMPBLOCK
		CPREVHASH
		CCURRHASH

How Counts May Be Taken

PHASE I

The following counts should be correct anytime before counts are combined during the input EOR routine:

Records written from current input reel, represented by X:
X = 9999 − AWRCNTX(2, 5) + AWRCNT

All records written this phase:
X + AWRITECNT

Records deleted from current input reel:
ADLTCNT

All records deleted this phase:
ADLTCNT + ADELETECNT

Records read (successfully) from current input reel:
X + ADLTCNT

Records read (successfully) this phase:
X + AWRITECNT + ADLTCNT + ADELETECNT, or
X + AINCOUNT + ADLTCNT

Hash total of records written from current input reel:
AOKHASH

Hash total of all records written:
AHASH

Hash total of records deleted from current input reel:
ADELHASH

Blocks dumped for error from current input reel:
IOCSFTBL01(8, 9)+4

All blocks of records dumped for error this phase:
IOCSFTBL01(8, 9)+4 + ADUMPCOUNT

PHASE II

The following counts should be correct anytime before counts are combined during end-of-pass procedures:

Records written this pass:
9999 − BWRITEX(2, 5) + BWRITECNT

Records summed this pass:
9999 − BSUMMX(2, 5) + BSUMMCNT

Records deleted this pass:
9999 − BDELETEX(2, 5) + BDELETECNT

Records written last pass:
BPREVCOUNT

Hash totals of records written this pass:
BHASHWRITE

Hash totals of records summed this pass:
BHASHSUMM

Hash totals of records deleted this pass:
BHASHDELET

Hash totals of records written last pass (or Phase I):
BPREVHASH

Blocks dumped for error from a given input reel:
Digits 8, 9 in the fourth word of the DTF for that reel

PHASE III

The following counts should be correct except during the output EOR routine and during the end-of-phase routine:

Sorted records written this output reel, represented by X:
X = 9999 − CWRITEX(2, 5) + CWRITEREEL

Inserted records this reel, represented by Y:
Y = 9999 − CINSRTCNTX(2, 5) + CINSRTREEL

All records (including inserted) written this output reel:
X + Y

Sorted records written this phase:
X + CWRITECNT

Inserted records written this phase:
Y + CINSRTCNT

All records (including inserted) written this phase:
X + Y + CINSRTCNT + CWRITECNT

Summed records this phase:
9999 − CSUMMX(2, 5) + CSUMMCNT

Deleted records this phase:
9999 − CDELETEX(2, 5) + CDELETECNT

All records written last pass of Phase II:
CPREVCOUNT

Hash totals of sorted records written on this output reel:

CHASHREEL

Hash totals of sorted records written this phase:

CHASHREEL + CCURRHASH

Hash totals of inserted records this output reel:

CHASHINSRT

Hash totals from last pass of Phase II:

CPREVHASH or CHASHWRITE

Blocks dumped for error from a given input reel:

Digits 8, 9 in the fourth word of the DTF for that reel

Current Record

ARECORD contains the RDW of the record most recently admitted to the scan of Phase I. BRECORD contains the RDW of the record of Phase II most recently moved to the bottom of the ranking; this record is PUT, summarized, or deleted. CRECORD contains the RDW of the record in the same circumstances in Phase III.

RDW Lists

In Phases II and III, the address of the RDW's for each file can be determined from digits (4 to 7) of the third word in the DTF for that file. In Phase I, an RDW describing the RDW list for reading is in index word AGREAD, and an RDW describing the RDW list for writing is the index word AGWRITE. The index words defining the current RDW list for processing and the spare list are modified by the internal sort, so the exact location of these lists is not so simply determined.

1. Index word AMTG contains the limits of the spare area until the internal merge begins, after which its indexing portion is incremented to control the placement of merged RDW's into this area.
2. At the beginning of the scan, index word ASCT contains the limits of the RDW list to be processed; during the scan its indexing portion is incremented to determine the processing loop. During the merge it has another function.
3. However, at the beginning of each merge pass, index word ASCTABLE is set to hold the first entry in the sequence table; thus, its indexing portion addresses the first location in the RDW list being processed throughout the merge pass.
4. Index word ASEQTABLE defines the *maximum* limits of the sequence table.

G

The bounds of the G's of Phase I are given in AGSIZE, AGSIZE +1, and, if a three-area system is used, AGSIZE +2. These words are located with the other constants unique to Phase I.

Other Locations

Other locations which may be found of aid in program testing are found in the same areas as the counters discussed above. These areas are the communications block common to all three phases, the constant areas unique to each phase, and the index words. The user may consult the listing for further information on these. The index words and their uses for Phases II and III are listed at the beginning of these phases.

Glossary

This is a glossary of basic sorting terms used in this manual. For a more extensive glossary of sorting terms see the *Appendix* of *General Information Manual, Sorting Methods for IBM Data Processing Systems,* F28-8001.

ASSIGNMENT PROGRAM: The set of instructions by which a generalized program modifies and completes the set of instructions which will be utilized in the performance of one specific machine run.

BLOCKING FACTOR: The number of data records contained within a tape record, or block. Sometimes shortened to "blocking."

BLOCK LENGTH or BLOCK SIZE: The total number of words contained in one block of records.

CHECKPOINT: A reference point at which error-free operation of the program has been verified and to which the program may return for restart in the event of subsequent failure. Checkpoint also refers to that routine in the program which writes the checkpoint record.

CHECKPOINT RECORD: A tape record of those contents of storage necessary to restart a program at the checkpoint.

COLLATING SEQUENCE: The relative order of precedence which a computer assigns to the numbers, letters, and special characters for compare operations.

CONTROL CARD: A card which contains parameters which, through interpretation by an assignment program, regulate the setting up of a generalized program for one particular application.

CONTROL DATA: Aggregate of all control fields in a record which are used for any one identification or sequencing operation on a file.

CONTROL DATA FIELD or CONTROL FIELD: A contiguous set of one or more characters in a record (not necessarily in the same word), on the basis of which, alone or with other control fields, an identification or sequencing operation can be performed.

CONTROL DATA SEGMENT: That part of a control field which is within a single word.

G: The set or the number of records ordered in storage by the internal sort.

GENERALIZED PROGRAM: A program which is designed to process a large range of specific jobs within a given type of application and which can compute instructions for itself so that it can perform one particular job. Present generalized programs have provisions to add and/or delete sections of instructions as they are required and to move areas within storage so that the running program is both compact and efficient.

HASH TOTAL: A total of data made for auditing or control purposes which would not ordinarily be added together, e.g., summing a list of past numbers. In multiple-pass computer applications (e.g., tape sorting), such hash totals are reconciled at the end of each pass with the hash total from the preceding pass. This provides a check for machine or program failures and may detect the presence of illegal characters in data.

INITIALIZATION: Resetting counters, switches, and instruction addresses at specified times in a program. This process is not to be confused with the assignment program which is performed only once, and is always executed before the running program has been started.

INTERNAL SORT: A sort in which all the records sorted into a single sequence remain in primary storage while the ordering is accomplished. The length of a sequence thus formed is limited mainly by the number of items to be sorted which will fit in storage at one time.

MERGE (verb): To combine items from two or more sequences into a single sequence. A common type of merging is to take several files, each of which is sequenced, and merge them into a single file, which contains all of the items from all of the files in a single sequence. The merging of two input files is said to be a two-way merge, three input files a three-way merge, etc. If the number of inputs has not yet been determined, the merge might be termed an M-way merge. M is also said to be the order of merge.

ORDER OF MERGE: The number of input sequences being merged (i.e., combined into any single sequence) during any one pass. The number of input files combined into a consolidated file during any one pass of a sort or merge.

PARAMETER: A quantity which is left unspecified at some stage of an operation and to which the user may assign arbitrary values.

PASS (1): A complete cycle of reading, processing, and writing an entire file.

PASS (2): One repetition of a repeated operation. For example, in this manual a cycle of the internal merge merges successive pairs of sequences until all records in a G have been processed. If all the records still do not form a sequence, another cycle occurs. Additional cycles occur until all the records in a G do form a sequence. Each cycle of this merge processes all the records in a G and is referred to as a pass.

PROCESS-LIMITED: The operating condition of a computer which exists when internal processing time exceeds input-output time. The term applies only to equipment which provides for overlapping, or simultaneous operation, of input-output and internal processing. The opposite condition is input-output limited, or, more specifically, tape-limited.

RECORD COUNT: The number of records in a file.

RESTART: The return to a previous point in the program to begin processing again. This previous point may be the beginning of the program or it may be a checkpoint. RESTART also refers to that routine in the program which accomplishes the return.

RUNNING PROGRAM: A generalized program which has been set up for a particular job by the assignment program with its control card parameters.

SEQUENCE BREAK: Condition when a record has control data which are lower-valued (in the collating sequence) than the previous record. The preceding sequence is concluded and a new one must be started. When sorting in reverse order, high to low, a sequence break is just the opposite from the condition described above.

SORT: To sequence or order a file of records according to some designated control data.

STEPDOWN: A sequence break.

TAPE LABEL: A record, usually at the beginning of the tape, ending of the tape, or both, designated for purposes of identification and control.

TAPE-LIMITED: The operating condition of a computer when tape input and output time exceeds processing time. This term only applies to equipment which provides for overlapping, or simultaneous input-output operations and processing. The opposite condition is process-limited.

Abbreviations

Addr	Address
Blkg	Blocking
Calc	Calculate
CD	Control Data
Ch	Chart
Chan	Channel
Chg	Change
Chk	Check
Chkpt	Checkpoint
Ctrl	Control
Cnt	Count
Cnts	Counts
Cntrs	Counters
Comp	Compare
Dr	Drive
Dup	Duplicate
EOF	End of File
EOR	End of Reel
Hlt	Halt
Incr	Increase
Instr	Instruction
IW	Index word
Lbl	Label
Loc	Location
Mk	Mark
Mks	Marks
No	Number
Ph	Phase
Prev	Previous
Proc	Process
Rcd	Record
Rd	Read
Req	Required
Rls	Reels
Rtns	Routines
Seg	Segment
Seq	Sequence
Spec	Special
ST	Sequence Table
Sw	Switch
Temp	Temporary
Wr-Rd	Write-Read

(10/61:7M-FW-72)

IBM 7070/7074 Sort 90 Printed in U.S.A. C28-6120

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York