CPB-1032

D. J. F.
STATISTICS
Section

COMPATIBLES/200

FORWARD SORT/MERGE GENERATOR

TV - C - Is Late printout?
Record & use Error Tape

GENERAL ELECTRIC

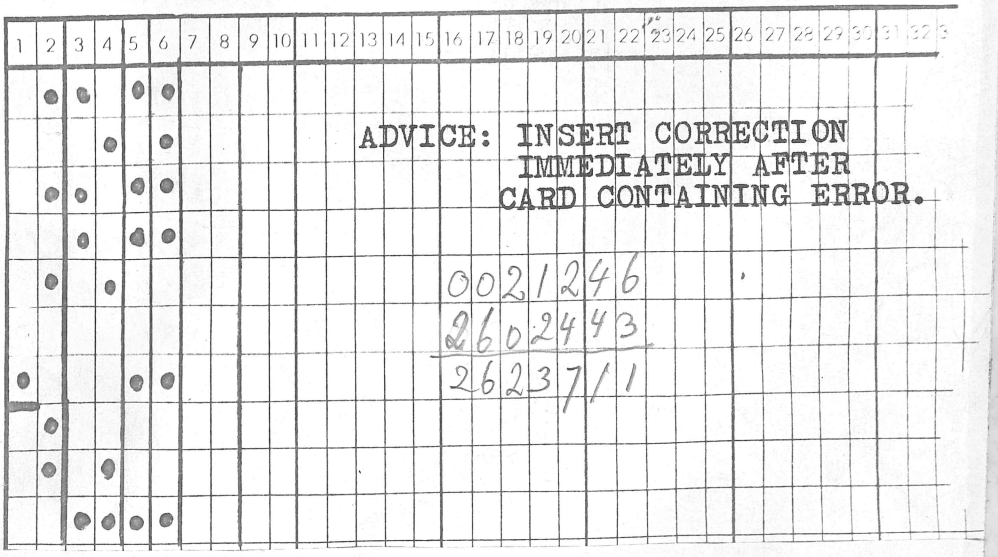
COMPUTER DEPARTMENT

CORRECTING
SORT BINARY DECK

OF THE BINARY CARD

WORD ONE (column 1,2)	THE FIRST 7 BITS CONTAIN THE NUMBER OF ACTUAL INSTRUCTIONS FOR MEMORY. THIS WOULD BE NUMBER OF WORDS ON THE CARD MINUS TWO. THE LAST 13 BITS CONTAIN THE STARTING LOCATION FOR THE INSTRUCTION(S).
WORD TWO (column 3,4)	CONTAINS THE FIRST INSTRUCTION. IT BELONGS IN LOCATION PUNCHED IN WORD ONE.
FOLLOWING WORDS	CONTAIN ALL FOLLOWING INSTRUCTIONS
LAST WORD	CONTAINS CHECK SUM OF ALL PREVIOUS WORDS AND IS CALCULATED AS FOLLOWS: WORD 1 WORD 2 + SUM IF OVERFLOW ADD ONE WORD 3 + SUM IF OVERFLOW ADD ONE ETC. UNTIL ALL WORDS HAVE BEEN ADDED TO GIVE CHECKSUM.

CARD LAYOUT



CORRECTION SAMPLE. (Where location 1246 is made to contain octal 2602443) checksum is 2623711

RERUN

I. INTRODUCTION

RERUN is a routine which will position magnetic tape data files and will restart a production program from a predetermined restart point. The restart point is basically a memory dump of the production program at a given time during its execution. The scheduling of a restart point must be planned by the programmer and be controlled from his program. In doing so, the programmer insures program recovery from a scheduled restart point rather than from the beginning if some unscheduled interruption should occur during the execution of his program. This ultimately results in effective utilization of the processing time allotted to the running of production programs.

II. MINIMUM SYSTEM CONFIGURATION

- 4K memory
- Magnetic tape subsystem with one handler
- Card reader

III. PROGRAM COMPATIBILITY

RERUN will recover programs having restart points created by:

- Symbolic Tape Input/Output System (SIOS) (CD225E2.001)
- FORWARD Sort/Merge Generator (CD225G1.002)
- GECOM Model Two Subroutine Library (CD225H1.001)

In addition, RERUN will reposition only the magnetic tape files associated with the program being recovered. The repositioning of files on other peripheral input/output devices must be planned by the programmer, except in the case of a GECOM object program. RERUN then types instructions to the operator to aid in positioning nonmagnetic tape files. The programmer must also reserve in his program at least 32 words at the upper limit of 4,096, 8,192, or 16,384 word memories. RERUN uses these locations to read the restart memory dump from magnetic tape.

For specific instructions to establish restart points in programs using SIOS or programs created by FORWARD and GECOM, the readers should refer to their respective manuals.

IV. PREPARING TO RESTART A PROGRAM

The RERUN routine is a self-contained binary program card deck. To use this program, the programmer must prepare a control card with the necessary information to initiate restart procedures. Before preparing the control card and subsequently using RERUN, the programmer should make certain that the memory dump from which recovery will be made was created according to the conventions of SIOS, FORWARD Generator, or GECOM.

RERUN Control Card

The RERUN control card provides the RERUN routine with information for locating the restart memory dump and restoring it in memory. All references to tape and label information pertain to the magnetic tape containing the restart memory dump.

The control card is prepared as follows:

Columns	Entry
1-15	Blank
16	Plug number
17	Tape number
18	Blank
19-21	Reel number (right justified)
22-30	File identification (program name, if special RERUN tape) of the tape containing the restart memory dump
31-36	File creation date
37	1 - if recovery is from a special RERUN tape; otherwise enter zero.
38-39	Blank
40-42	Identification number of the restart memory dump on special RERUN tape; otherwise leave blank.
43-80	Blank

The above entries are related to file identification and labels, and may be obtained from the Console Typewriter Log at the time the restart memory dump was created on an intermediate output tape. If a special RERUN tape is used, the program name and data may be obtained from the Run Operation Sheet. The reel number and identification number may be obtained from the RPXXX number on the typewriter.

V. CONVENTIONS

In addition to locating the restart memory dump, RERUN will automatically position magnetic tape data files used by the program being recovered. If the restart is established by SIOS or FORWARD, RERUN will position a maximum of 30 files; and if established by GECOM, a maximum of 16 files.

VI. OPERATING INSTRUCTIONS

RERUN is a two-packet binary program card deck. The RERUN control card must be punched and inserted immediately after the transfer card of packet No. 1. Packet No. 2 is an overlay of the RERUN program and must be placed immediately after the control card.

Computer Setup

1. Mount the tape reel containing the restart memory dump on the magnetic tape handler number and controller selector plug number given on the RERUN control card.

2. Mount all other magnetic tape data files used by the program about to be restarted. These reels must be in a rewind position.

3. Position data files on other peripheral input/output devices used by the program.

4. Place the RERUN binary program card deck (with the control card inserted between packets 1 and 2) in the input hopper of the card reader. Place two blank cards after this program card deck.

5. Follow standard instructions for loading a card program deck.

Normal Console Typewriter Messages

The message

RPXXX

indicates recovery has been initiated. XXX is the identification number of the restart memory dump if the dump is contained on a special RERUN tape produced by GECOM.

If the program being recovered is a GECOM object program, RERUN will type the following repositioning information for files other than those contained on magnetic tape.

Message	Meaning
CDP XXXXXX	Indicates the number (XXXXXX) of cards that were punched at the time restart memory dump was established.
HSP	Informs the operator to set the printer to the state that existed at the time the restart memory dump was established.
CDR XXXXXX	Indicates the number (XXXXXX) of cards that were read from the card reader at the time the restart memory dump was established. (The operator must position the card reader file to read card XXXXXX + 1.)
OPR	Informs the operator to set up files for other input/output devices to the position that existed at the time the restart memory dump was established. The RERUN program enters a halt loop to permit the operator to set up these files. To continue recovery, the operator must toggle console switch 0.

RERUN will type the messages listed below for each magnetic tape file used by the program being recovered. It will then enter a halt loop to permit the operator to mount the tape if he has not already done so.

If or when the file is mounted, the operator must toggle console switch 0 to continue. This process is repeated for each magnetic tape file.

When all files required by the program being restarted have been properly mounted, RERUN will type

RPXXX

to indicate that recovery is completed and execution of the program has started.

The following are messages pertaining to mounting (or previously mounted) magnetic tape files used by the program being restarted.

Message	Meaning
Px Ty (tape label)	Plug (x) and tape (y) numbers and the tape label.
DUPLICATE Px Ty MOUNT PER RUN OPERATION SHEET	Indicates the program uses the same magnetic tape handler for more than one file, and the operator must mount the file that was being used at the time the restart memory dump was established.

For files not having labels, RERUN will type the plug and tape numbers and one of the following messages:

Programs using SIOS or FORWARD:

NO LABEL Px Ty
MOUNT PER RUN OPERATION SHEET

GECOM Object Program:

nnS Px Ty
MOUNT PER RUN OPERATION SHEET

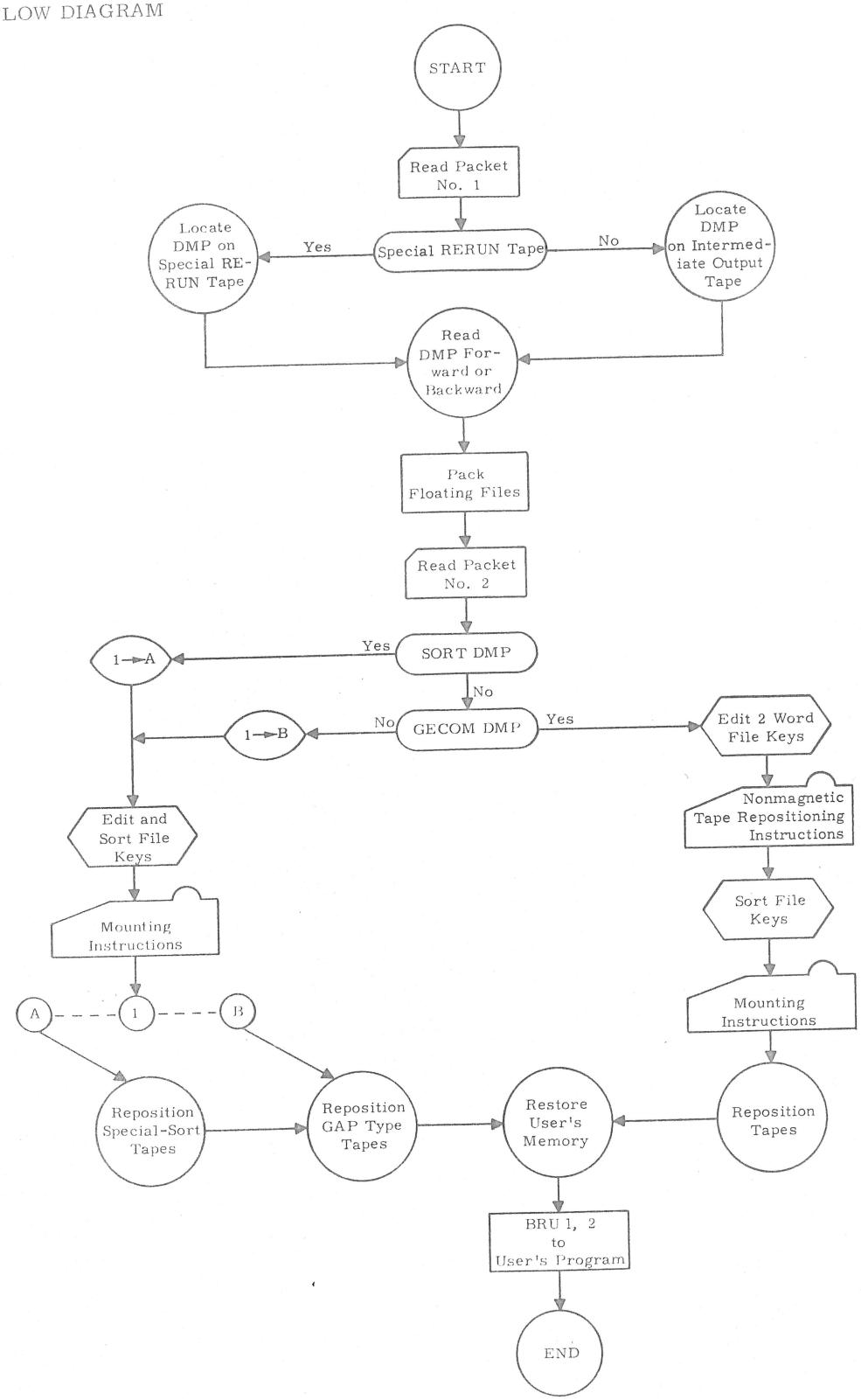
where nn is the file number that GECOM assigned to the file.

Error Messages

Message	Meaning
Px Ty (tape label) NO MT (expected tape label)	Occurred when positioning the tape containing the restart memory dump. The tape label information given on the RERUN control card does not agree with the tape label of the tape containing the restart memory dump. "Tape label" is the identification found on the tape; and "expected tape label" is the information read from the control card. ACTION: Determine whether the control card contains the correct information. Correct and restart RERUN from the beginning. Determine if incorrect tape was mounted. Mount correct tape and toggle console switch 0 to continue. The incorrect tape may be accepted by setting switch 19 and toggling switch 0.
Px Ty (tape label) NO MT (expected tape label)	Occurred when positioning a data tape associated with the program being restarted. The file identification found in the file parameter list does not agree with the file identification on the tape label. "Tape label" is the identification found on the tape; and "expected tape label" is the identification found in the file parameter list or GECOM I/O Table. ACTION: Determine if incorrect data tape was mounted. Mount correct data tape and toggle console switch 0 to continue. The incorrect data tape may be accepted by setting console switch 19 and toggling switch 0.
E2	The file being positioned has a block size of more than 500 words (4,096-word memory) or 1500 words (8,192- or 16,387-word memories). RERUN halts. Conventions for establishing a restart point were not followed. RERUN cannot recover this program.

Message	Meaning
E3	A restart cannot be accomplished using this memory dump. Prepare a new control card for the previous restart point and reload the RERUN program.
Px Ty E4 Px Ty E5 Px Ty E6	A tape parity, Mod 3/4 or I/O buffer error occurred and RERUN was not able to override it.
EOF	Desired memory dump is not on this special RERUN tape. Mount another reel, toggle switch 0, and RERUN will proceed to search the new tape for the dump.

VII. FLOW DIAGRAM



RERUN

GE-215 / 225 / 235

FORWARD SORT / MERGE GENERATOR

PROGRAM NUMBERS
CD225G1.002
CD225G1.006

41.6 (4.4) TAPE SORTING TIME SCHEDULE

JULY 1962
REVISED MARCH 1964

GENERAL ELECTRIC
COMPUTER DEPARTMENT

RFL
END TAPE LABEL

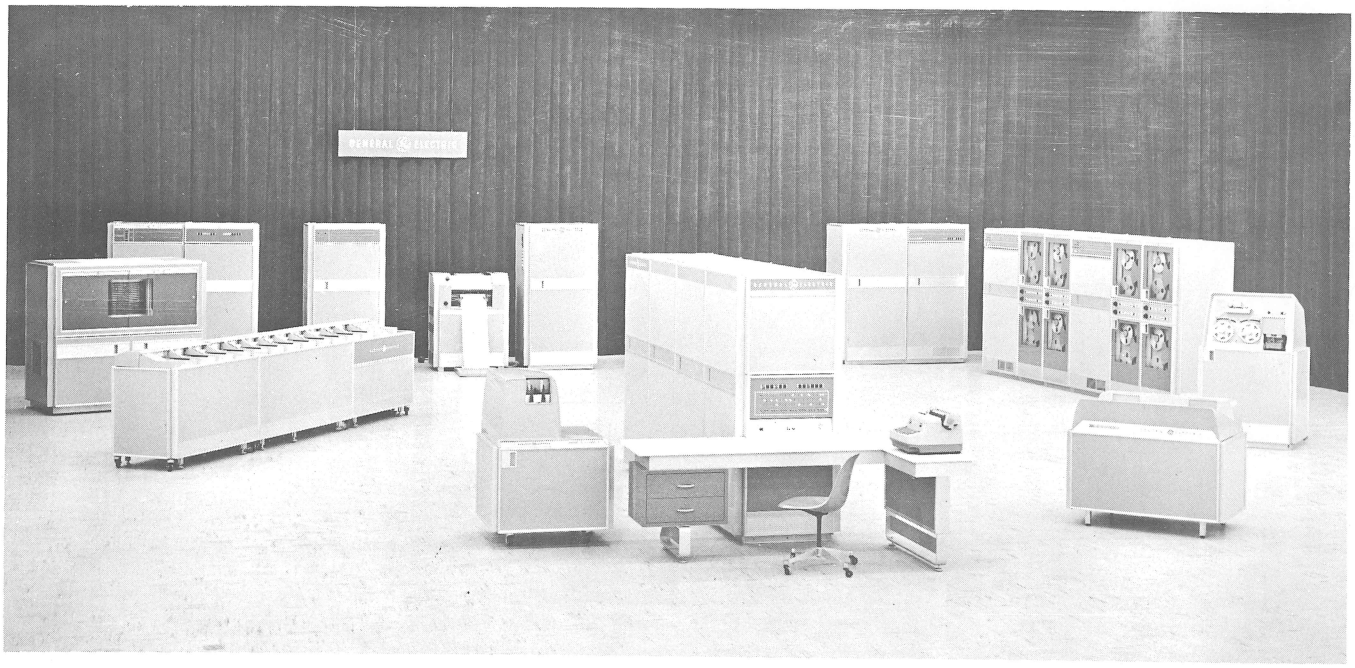
BTL
END TAPE LABEL

BTL
END FILE LABEL

FOR MULTI INPUT.

CONTENTS

ILLUSTRATIONS



I. INTRODUCTION

The FORWARD Sort/Merge Generator (CD225G1.002) and the FORWARD Sort/Merge Generator-8k-BRIDGE II Compatible (CD225G1.006) are generalized GE-215/225/235 program generators which produce sort and merge programs. The generated programs are tailored to specific applications on the GE-215/225/235 Information Processing Systems. Although the first ten chapters of this manual indicate that they are written for the generator CD225G1.002, the information applies equally well to both generator programs. Chapter XI contains amplifying information specifically for the CD225G1.006 program. It is advantageous for any user who has an 8k or larger memory to use the BRIDGE II Compatible program. Sort and merge object programs can be placed on magnetic tape using BRIDGE II and can be operated from a conventional instruction tape. The system tape can be updated and runs can be sequenced under BRIDGE II control.

In the remainder of the text, the term FORWARD will be used as an abbreviated title for either of the FORWARD Sort/Merge Generators. FORWARD (File Order Rearrangement With Ability to Revise Data) generates any desired object program for sorting or merging data files according to descriptive parameters specified by the user. In addition, the generator accepts General Assembly Program symbolic coding prepared by the user for inclusion in the sort or merge object programs to process input files, to combine or eliminate duplicate data records, and to process output data as its final position in the output is established. Complete flexibility is thus attained in data selection, data format arrangement, and the use of input/output media other than magnetic tape.

References to the General Assembly Program actually refer to three different programs, and the reader must identify the specific program according to the following:

1. General Assembly Program Model Two Systems Tape - 8k (CD225F1.007) is used with FORWARD (CD225G1.002) in an 8k system.

2. General Assembly Program Model Two Systems Tape - 4k (CD225F1.010) is used with FORWARD (CD225G1.002) in a 4k system.

3. General Assembly Program II Systems Tape - BRIDGE II Compatible is used with FORWARD (CD225G1.006).

Both FORWARD Sort/Merge Generator programs are continually updated to reflect any changes made to the Symbolic Tape Input/Output System (CD225E2.001).

Most data files are composed of records arranged according to a sequential key. For magnetic tape applications, sort and merge programs must be used to arrange the records in sequence according to the key and to place the records in proper position in the sequence. Record sequence usually is designed to have a convenient relationship to the desired reporting order. Updating of sequential files is relatively easy where the transaction records have been rearranged in the same sequential order as the file being updated.

Sort programs arrange the records of a data file in sequential order with respect to their sequence key values. Sort programs can produce the initial ordering of a master file, arrange transaction records in order for a subsequent updating program, correct minor sequence deviations in a nearly sequenced file, and arrange records into a special order for reporting. The input file is normally assumed to be arbitrarily disordered. In a series of magnetic tape manipulations, the records are redistributed and gathered together until they are finally delivered to the output file in the desired order. If the input could all be held in the computer memory at one time, no recourse to magnetic tape would be required, and sorting could be done entirely by internal key comparisons and record moves. This is sometimes possible for small amounts of data. For such applications, refer to CD225C5.001, Internal Memory Sort. The FORWARD sort is capable of handling very large amounts of data. The class of problems considered in this publication is characterized by the assumption that all of the data being sorted cannot be held in memory at one time.

Merge programs combine several sequential input files into a single sequential output file. Merges may be used to reduce several newly sorted files to a single file, or to match transaction records from several files sequentially against a master file.

Preparation of input for the FORWARD Sort/Merge Generator requires only an understanding of the files to be processed and results desired; the user does not need to concern himself with the complex logic of sort and merge programs. The user is relieved of the need to plan elaborate processes to merge large number of files; the planning is designed and executed by the variable-merge object program. FORWARD not only saves the user many weeks of programming effort, it also saves him many hours of computer time. For example, for a minimum application the user may fill out four descriptive parameter cards and submit them to the FORWARD generator. The generator then takes only about two minutes to write a complete 2500 instruction sort program on magnetic tape in the form of symbolic instructions for input to the General Assembly Program.

Parameters, punched into control cards by the user, initialize the generator and preset the record sizes, key sizes, and blocking factors, User Coding Elements may be prepared in General Assembly Program language on punched cards to handle varying input and output formats or media, to combine or eliminate records having the same sequence key, or to use non-standard collating sequences. The generated sort or merge object programs are fully optimized with respect to use of optional instructions, comparison routines, data moves, magnetic tape blocking, memory assignments, and available magnetic tape handlers.

FORWARD sort and merge programs generated with only the descriptive parameters included will process magnetic tape data files with the following characteristics:

Individual record sizes ranging from 1 to 999 words.

Magnetic tape record blocks ranging from 24 to 999 words. (The record blocks range from 1 to 999 words where the tape handlers are equipped with the vacuum-pocket tape feed.)

Any number of reels of magnetic tape.

An arbitrary number of input records with multicycle operation.

Conventional labels and fences. (See CPB 178, GE-225 Programming Conventions.)

Sequence key occupying 1 to 99 words.

Sequence key occupying as many consecutive full words as necessary at the beginning of the records. The key may be scattered throughout the first 30 words of the record. Key words for BCD data must be positive values. For binary data, negative values are acceptable.

Priority program execution is parallel with sort and merge object programs through Automatic Priority Interrupt feature.

The System hardware configuration minimum requirements are:

FORWARD sort or merge program generator.

GE-215, 225, or 235 central processor with 4096 word memory.

3 magnetic tape handlers on one controller.

1 card reader.

1 high-speed printer.

FORWARD sort or merge object program.

GE-215, 225, or 235 central processor with 4096 word memory.

3 magnetic tape handlers on one controller. (4 magnetic tape handlers are required where the multicycle feature is used.)

1 card reader.

System hardware configurations are extremely flexible for FORWARD sort and merge object programs. The minimum system configuration may be expanded to include the following:

Memory size up to 8192 words.

1 or 2 magnetic tape controllers.

Move (MOV) instruction option may be used, if available.

Compare (CAB and DCB) instruction option may be used, if available.

Automatic Priority Interrupt (API) option may be used, if available.

Memory size of 16,384 words with upper 8k used for User Coding Elements.

User Coding Elements may be included in FORWARD sort and merge programs to permit the system to process any input or output formats or media. When present, they become an integral part of the tailored object program. User Coding Elements may be used to process:

Punched card input or output.

Paper tape input or output.

High-speed printer output.

Variable size input or output magnetic tape record blocks.

Input records, where the sequence keys are not compact, or where the keys are not in the first words of each record.

Multifile tapes.

Extra input or output files.

Sequence keys in which numeric digits are assigned greater magnitude than alphabetic characters, or in which special character sequences are assumed.

API OBJECT PROGRAM

Automatic Program Interrupt (API) object programs allow the user to execute priority programs in parallel with sort and merge programs in response to hardware interrupts occurring at any time except during input or output and any other real time processing. A generator input parameter is made available in the program parameter card to select the API option.

When the API option is selected, the FORWARD generator supplies the Automatic Program Interrupt (API) Executive program (CD225J4.000R) in the object program. The instructions for using the Symbolic Tape I/O System (CD225E2.001) with automatic priority interrupt should be consulted if the user wishes to set up his own parameter tables in Input Coding Element (ICE) or Output Coding Element (OCE). Priority programs using magnetic tape must use the Symbolic Tape I/O System (SIOS) subroutine supplied in the sort or merge object program. The writeup of the API Executive program should be consulted for conventions referring to automatic priority interrupt.

The API Executive program begins in decimal location 166 (octal 246). The sort/merge object program initializes the API Executive program. The interrupt mode is on except when the input/output functions of the object sort or merge program are being executed. The priority interrupt program(s) should be in memory at all times. In a sort object program where there are two subprograms which are not in memory at the same time, the user must origin any interrupt program in the #CS or #SCE area. Locating the priority interrupt programs at these symbolic locations insures that they are always in memory regardless of whether the presort or the rotary-merge subprogram is being executed. At the end of a sort or merge object program, a branch is made automatically to the end of job subroutine of the API Executive program to permit the priority interrupt programs to finish processing.

Coding Form. Column 16 of the program parameter card has been set aside to contain a parameter to implement the API feature into sort or merge object programs. Two valid entries may be punched in column 16:

An N punch or blank in column 16 signifies that API is not to be implemented into the sort or merge object program.

A Y punch in column 16 causes the FORWARD generator to produce the object program with the API feature plus the API Executive program. Any other punch in column 16 is considered to be an error, and no API feature will be generated.

Conventions. The sort object program is considered the main program under API convention. The main program follows the conventions listed in the API Executive writeup (CD225J4.000R). The API initialize routine is executed before starting a presort or merge program. An appropriate "end of job" routine is executed at the end of the sort or merge program. When a run completion routine is present, the main program disables the API before going to the next program. Otherwise, the main program's ending function is performed according to the API Executive program convention.

Priority programs require origin zero cards as designators and are loaded with the main program under control of the General Absolute Loader II (CD225B1.013R) or equivalent loader with an API module. Priority programs must also meet the conventions and specifications as stated in the API Executive program including appropriate starting and ending functions.

Operating Considerations. Entrance to priority programs must be located in lower 8k. This entrance may provide linkage to and from the body of a priority program which could appear in upper 8k. It is only necessary that the object loader and the API Executive program find the priority entrance in lower 8k of memory. As described in the paragraphs on conventions, either a static or dynamic load may be obtained. The API Executive writeup describes operating considerations for these two situations.

Object programs are loaded by the General Absolute Loader II or by a loader with the same characteristics and conventions. The priority programs or their indicators (ORG 0) must precede the main program at load time. Transfer cards must be removed from priority programs and the cards must be placed in the order in which they are to be executed if a desired sequence exists.

Index group 32 is selected during interrupt, and registers 1 and 2 are used by the API hardware and the API Executive program respectively. These index registers may be saved and restored if their use is needed. The API initialize routine of the API Executive program builds a table of priority programs to be executed from the VECTOR table supplied by the loader and provides proper linkage between location $(132)_{10}$ and the API Executive normal entrance.

II. DESCRIPTION OF FORWARD SORT OBJECT PROGRAM

The FORWARD Sort/Merge Generator produces optimized sort programs to process any GE-215/225/235 data file. The generator output is a complete object program with all necessary subroutines and input/output procedures built in. The output is written on magnetic tape as a symbolic input to the General Assembly Program. The user only adds loaders to put his generated and assembled sort program to work.

Each FORWARD sort object program consists of a presort subprogram and a rotary-merge subprogram. The presort subprogram arranges as many input records as possible into sequences (called strings), and writes each string onto a collation tape. The rotary-merge subprogram progressively collates the strings together until all of the data is written in a single string onto an output tape. The presort subprogram processes all of the input data in a single pass. The rotary-merge subprogram then processes all of the data in one or more (usually several) additional passes. The total number of passes required depends upon record size, number of records in the file, sequence of the input, memory size, and number of collation tapes. All of these factors affect the number of records written in each string (string length). For a given volume of data, the greater the string length, the fewer the strings; and the fewer the strings, the fewer rotary-merge passes (phases) required. Therefore, less total time will be taken to run the sort. The objective during the presort is to maximize the initial string length and minimize the number of strings.

The presort uses a large block of memory between the program (low addresses) and the buffers (high addresses) for "tournament storage." The presort reads as many input records into tournament storage as can be held, and keeps this area full until the input file is exhausted, replacing each record removed for output with a new input record. Maximizing the size of the tournament storage is essential for efficient sorting. Its size is inversely proportional to the number of strings the presort is expected to produce.

The word tournament characterizes the key comparison process used throughout FORWARD. The number of comparisons required to find the least key record is considerably less than the number of records (n). In fact, the number of comparisons required to find the least key among n records is approximately $\text{Log}_2 n$. For example, to select the least key record from 100 records, just 7 comparisons are needed, as shown in the following table:

Comparisons Required	Records
1	$1 \leq n \leq 2$
2	$3 \leq n \leq 4$
3	$5 \leq n \leq 8$
4	$9 \leq n \leq 16$
5	$17 \leq n \leq 32$
6	$33 \leq n \leq 64$
7	$65 \leq n \leq 128$

The tournament routine is the nucleus of the presort subprogram. It selects the "winner" (lowest key record) from all the records in tournament storage. This record is then sent to the output string and replaced with a new input record. Continual reference is made to input and output routines to remove and replace winners. This enables every FORWARD sort to make effective use of the system's capability to compute simultaneously with read and write, since alternating buffers are always assigned.

The key of the new input record is compared to that of the winner it replaces. If the new key is less than that of the winner, the record is "marked" for the next string, but if it is not less, it is "marked" for the current string. The tournament is then re-entered. "Current string" records continue to win until no more are contained in tournament storage. As long as new records with high keys keep coming up, the current string is continued. When next string records fill tournament storage, one of them will win; the presort thus detects a "string break," and continues its work, with the former "next string" records now building the current string. Since records with ascending keys can continue to enter the current string indefinitely, there is no inherent maximum string length in any FORWARD sort. The minimum string length is the number of records that can be held in tournament storage; the expected string length is twice the minimum (for random input). If the input is "favorably biased" (i.e., the input tends to be in the desired order), this factor will be used to a good advantage, producing longer strings than for the same file in random order. As the tournament storage receives "next string" records, they are compared to each other in the course of "current string" comparisons, although only a "current string" record can win. Therefore, there is no pause for setup processes at a string break. The tournament will have already begun to establish the order for a new string, while selecting winners for the "current string."

When all the input has been read, most of the records will have been passed out to the collation tapes, but the tournament storage will still be full. The presort continues to substitute high-key "dummy" records as each winner is removed. When a dummy wins, all the records are out. Then the presort closes its collation tapes and brings in the rotary merge, which overlays most of memory with program and buffers. Although all of the records will have passed through the main processor, sorting is by no means finished. The operator may find it necessary to remove the input tape at this point and replace it with a blank magnetic tape before starting the rotary merge.

The rotary merge builds a single string on its output tape by combining a string from each of the other collation tapes. These strings are not tied end-to-end but are woven together. When the rotary merge has exhausted a string from each of the collation tapes, it begins building a new output string. Thus each output string reduces the number of strings on each input tape by one.

The presort deliberately writes a different number of strings on each tape stacked so that each tape will run out when the rotary merge has proceeded for a certain time. At this point, the rotary merge rewinds the exhausted input tape and the current output tape, and switches their functions. It continues processing the remaining strings on the other input tapes, while also merging from the tape it has just completed writing. Its output is now "exhausted input" tape. When the end of another input tape is reached, this process is repeated. The presort has written out strings in such a way that the input tapes to the rotary merge will be exhausted one at a time. All but one of the collation tapes are always being merged onto the remaining tape. A "phase" is completed each time an input tape is exhausted. The rotary merge continues through as many phases as are necessary to arrange the records into a single string. From 60% to 75% of the records being sorted are read and written during each rotary-merge phase, and 100% during the last phase. (See Figure 1.)

If over one full reel of data is given to the sort at one time, the collation tape handlers may have to write more than a single reel in each rotary-merge phase. The resulting tape changing costs a substantial amount of time. It is highly advisable to avoid this possibility by either using the multicycle feature or by planning to sort each reel of a multireel input file separately, and to follow the sort with a FORWARD variable merge to gather the entire file into the final output sequence. Mechanisms are included in the sort object program to enable the operator to force the ending of input at the end of each reel, to force continuation of the input at the end of each file, and to maintain unique output reel numbers in a multireel sort. (For details see Section IX.)

The purpose of the recommendation of a single reel limit is to avoid collation tape swapping. Although the object program actually permits collation tape swapping, the practice leads to operator confusion and consequent errors and wasted computer time. Appendix A (at the end of this manual) presents a method of determining how many input reels can be used with various block sizes without requiring swapping of collation tapes.

FORWARD sort object programs process conventional GE-215/225/235 tape files in either binary coded decimal (BCD), 18-bit special binary, or 20-bit binary mode. Blocked records (several logical records per physical tape block) are accommodated as long as logical record length is a fixed number of words. Variable record lengths may be handled automatically as long as only one logical record is contained in each input block. The sort record length should be set equal to the maximum input record length. The results are as follows: The sort issues tape read commands specifying the indicated record length; it takes the entire input buffer contents as the record to be sorted, carrying whatever "hash" is left in the buffer at the end of short input records. Thus, it actually writes fixed logical record lengths on its intermediate and final outputs. Subsequent programs can then ignore the hash carried at the end of each short record on the sorted file. Blocked variable-length records must be processed under the control of an Input Coding Element.

An Input Coding Element (ICE) should be provided whenever the standard assumptions about the input do not hold; that is, when the file contains variable record lengths, variable block lengths, or several record formats, or when the key is outside the range automatically treated, or when a special collating sequence is desired. (A sort can automatically process any key occurring in the leading consecutive whole words of the record in order of decreasing significance, or a scattered key of up to 15 whole word pieces occurring in any order in up to the first 30 words of the record.) ICE may also select or edit records, produce reports on the input file, and take input from a non-tape medium to master-file processing. The programmer may indicate by parameter options whether ICE contains its own input routine or whether the sort should obtain the input records independently and execute ICE as an editing routine. Overall control (outside of beginning and ending) may be maintained by either ICE or the presort subprogram, at the programmer's convenience. ICE takes memory that would otherwise be used for tournament storage. If it uses as much as half of such memory, it is almost certain to add a rotary-merge pass, as well as add the time to perform the ICE. Consideration should be given to a separate prior program instead of ICE in this case, for the sake of optimum overall timing.

The FORWARD multicycle feature may be used when it is expected that the number of records in the sort object program output may exceed the capacity of one magnetic tape reel. The multicycle option provides the capability in a generated sort object program to interrupt input procedures, sort the records, and resume input procedures each time an arbitrary number of records has entered the sort. An MCYCLE parameter card may be supplied to the FORWARD generator to specify the number of records to be sorted in each cycle, along with the priority channel and tape number of the magnetic tape handler reserved for multicycle use. This tape handler must not be used for any other purpose in the sort object program.

When the stipulated number of records has entered the sort, the rotary-merge subprogram is executed, producing a sorted tape. After the rotary merge has finished, it halts to permit the sorted output reel to be removed and a blank magnetic tape reel to be mounted. The presort subprogram is then automatically called back into memory from tape, and the input procedures continue until another cycle has been completed or all the records in the input file have been exhausted.

The cycle limit specified in the MCYCLE card should be sufficiently small, in view of record size and output blocking factor, to assure that the amount of data sorted in each cycle does not exceed one full reel of magnetic tape.

The output of each cycle of a multicycle sort is written as a separate output file, normally on a single reel of magnetic tape. A variable merge program should then be used to reduce the multicycle sort outputs to a single merged output file. If the Output Coding Element (OCE) is used, it should, in most applications, be incorporated into the variable merge rather than into the multicycle sort, since OCE is intended to operate on the final output file. If OCE is used in a multicycle sort, it operates separately on the output of each cycle rather than on the final output file. The programmer may think of the presort as a single continuous pass--as if the multicycle interruptions did not occur at all.

If User Coding Elements are used in a sort employing the multicycle feature, the following table should be consulted to ascertain when the user's own coding will be executed.

User's Coding Element	When Executed
ICE Starting	Once, at the beginning of the sort program.
ICE Ending	When the input file has been exhausted.
ICE Normal	On every input record.
OCE Starting	In each cycle at the start of the last rotary-merge phase.
OCE Ending	In each cycle at the end of the last rotary-merge phase.
OCE Normal	On every record during the last rotary-merge phase of each cycle.

Symbolic location #MCS contains the cycle number of the current cycle; it is initially set to one and is incremented by one each time the presort is resumed after a multicycle interruption. Symbolic location #MSEND may be tested by the user to determine if the current cycle is the last cycle. If #MSEND is non-zero, the last cycle is being executed.

An Output Coding Element (OCE) may be written to revise formats and records and to select and handle special types of output data as the final output sequence is established in the rotary merge. OCE may also produce printed reports and write the output on a medium other than magnetic tape. The presence of OCE in the FORWARD sort object program always causes the rotary merge to execute at least one pass. OCE is not executed until the last rotary-merge phase; that is, until the entire file is finally being arranged into a single string. For a multireel

input, OCE should normally be included in the variable merge which is generated to follow the sort, rather than in the sort itself. Like ICE, OCE takes memory that would otherwise be used for other purposes to increase efficiency. In rotary merge, all otherwise unassigned memory is used for buffers for the collation tapes (two buffers per tape). OCE results in smaller block sizes on the collation tapes and an increase in overall tape time. A large amount of OCE can be quite costly.

The absolute limits on memory available to ICE and OCE are much larger than the practical limits. Imaginative use of these features can save considerable amounts of effort outside the sort. But the chief reason they are provided is to extend the generality of FORWARD beyond tape files with fixed record sizes and formats, to allow the use in a sort of any source and final storage media and any data formats desired. Normally it is best to restrict ICE and OCE to editing, selecting, and medium conversion necessary to the sort problem itself. There is little danger of approaching the memory limits, either practical or absolute, when this rule is observed.

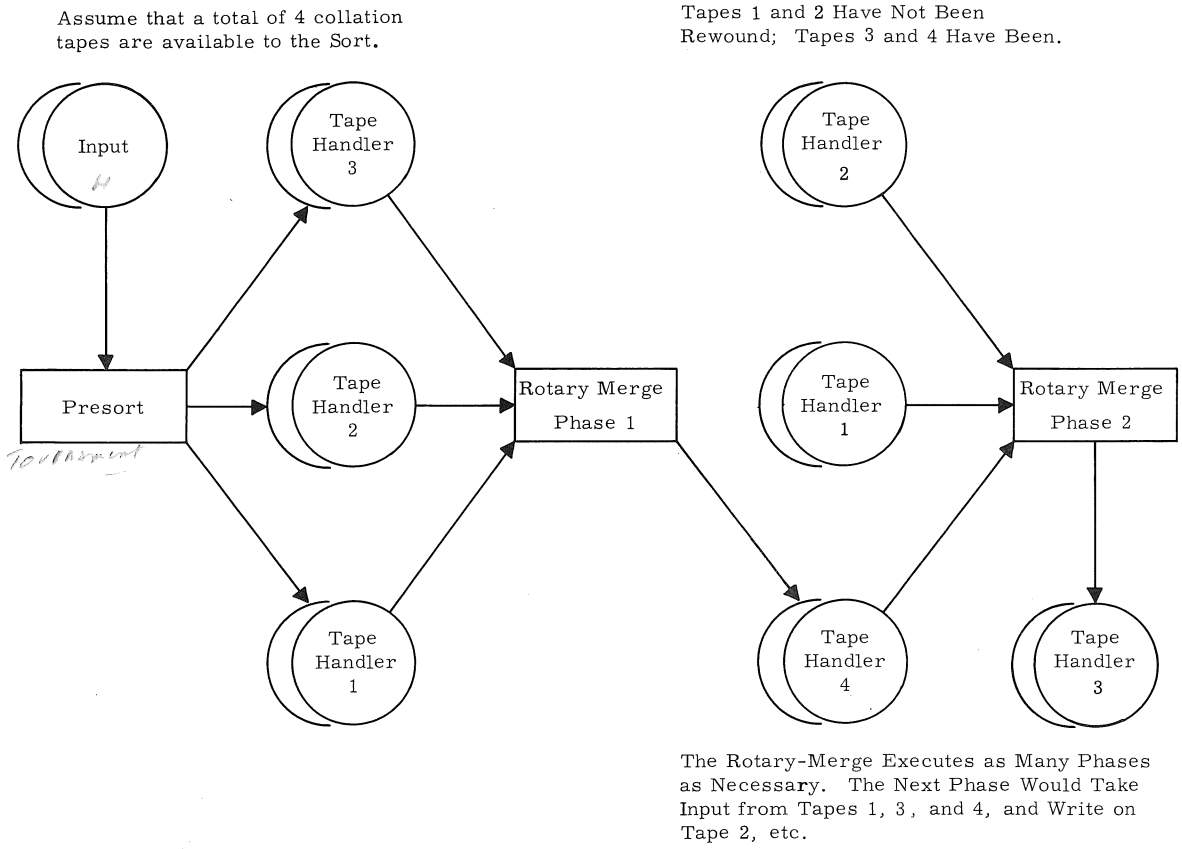


Figure 1. Flow of FORWARD Sort

A third type of User Coding Element--Squeeze Coding Element (SCE)-- may be provided when it is desired to combine or eliminate records having the same sequence key value. It will be executed whenever two records having the same sequence key are encountered, whether in the presort or the rotary merge. SCE may keep either or both records, but may not eliminate both. Unless the two records are identical, a decision to eliminate a particular record should be based entirely upon information contained in the records. SCE will be of marginal importance unless it is expected that a considerable percentage of the file may be eliminated. (See Figure 2.)

An example of the effective use of SCE is the case where the input file consists of a large volume of records, each containing an account number and an amount field. The desired output of the sorting operation is a file or a list of summarized records in sequence by account number. The input may contain, in random order, many records with the same account number, but the output must contain only one summary record per account number. The SCE option permits this kind of summarization while the sorting process is performed. Total sort time can be reduced when the expected data volume reduction by summarization is significant.

No assumption may be made about the order in which records with identical sequence keys are presented to either SCE or OCE.

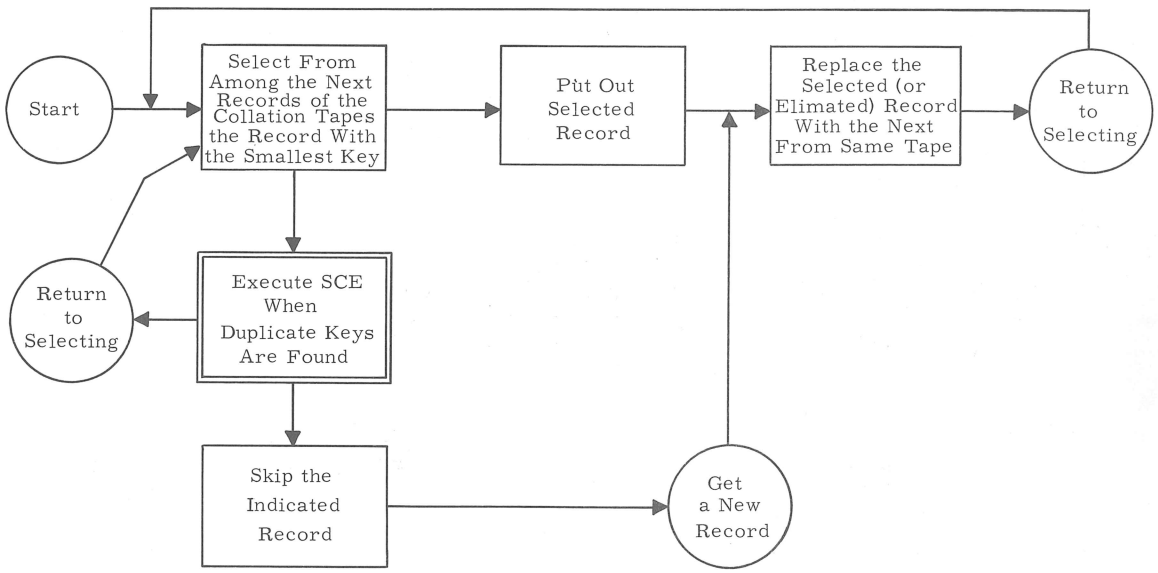


Figure 2. Flow of Rotary Merge with SCE

The user may communicate between the various User Coding Elements or with other programs, executive routines, and subroutines by reserving an arbitrary amount of upper memory (lower 8k) for communication storage. This area is fixed at the time of program generation, and is ignored and never destroyed by the sort program. A maximum number of 1999 memory locations may be reserved in systems having a 4096-word memory and a maximum of 3999 locations in systems having 8192- or 16,384-word memories. Efficiency is greater when fewer locations are reserved for communication storage.

The constants in the table which follows are reserved for special input/output purposes or for sort control indicators. They must not appear in the indicated positions in ordinary data records.

Octal Constant	Word Position	Special Purpose
0777777	1st key word	String break control in BCD mode.
1777777	1st key word	String break control in binary mode.
2777777	1st word of physical block	End-of-tape or file in Input/Output.
3777777	1st word of a record	Short data block indicator in Input/Output (binary mode).
0537453	1st word of a record	Short data block indicator in Input/Output (BCD mode).

When each sort intermediate output tape or file is closed, a check point for rerun is established by writing a memory dump after the data. Thus a rerun point is established at the end of each pass, so that the current pass may be restarted if necessary. It is also possible to suppress the memory dump for reruns when the user wishes to eliminate the rerun function. Suppression of the dump is an advantage when object programs process small amounts of data. Since the memory dump for rerun is recorded after the end-of-tape reflective marker, those installations whose magnetic tapes have less than 100 feet of tape after the reflective marker may wish to suppress the rerun memory dump to prevent overrunning the tapes.

Suppression of the rerun memory dump at the end of sort object program tapes is controlled by an entry in column 15 of the output parameter card. If column 15 contains a "Y" or is blank, the object program will dump memory at the ends of output tapes for rerun purposes, and will type the conventional "PX TY label RP" message when each dump is written. If column 15 of the output card contains an "N," no memory dumps will be written and consequently no "RP" messages will be typed. (See Section IX for operating instructions for FORWARD sort object programs.)

III. DESCRIPTION OF FORWARD MERGE OBJECT PROGRAM

A FORWARD merge program can be generated to collate any desired number of files, with any number of magnetic tapes for each file. Generated merge programs fall into two categories: fixed merge and variable merge.

Fixed merge object programs may be generated when it is known that the same number of input files will always be merged, and when enough tape handlers are available to provide at least one for each input file and one for the output file. The user provides a list of his assignments to the generator and he may associate each file label with one or two tape handlers.

If the number of inputs varies or if not enough tape handlers are available for a fixed merge, a variable merge may be generated. The user provides a list of all available tape handlers to the FORWARD generator, and each time the merge object program is executed it assigns tape handlers and optimizes itself for the actual input. (See Figure 3.)

When a variable merge program is to be executed, the number of files to be merged is specified by the operator by use of the console switches. During execution, the merge object program causes operator instructions to be typed out on the console typewriter. Since rewinding of tapes is faster than reading, a multicycle merge program runs much faster if extra tapes are used for additional input files rather than tape swapping. Therefore, no input tape swaps are provided in a variable merge. The variable merge does not check labels unless the user provides a Label Coding Element (LCE) to generate expected input file labels. If the LCE requires a card input, an area should be reserved using #CA, indicated on the output card. If desired, tape swap may be used for the output file.

Record size, sequence key, blocking factor, and tape storage mode (binary, special binary or BCD) are assumed to be constant and identical for all input files in the FORWARD merge object program. If sequence keys are not identical in format or in position in the input record, the programmer may write an Input Coding Element (ICE) to preprocess the input records as they are presented to the merge program. When ICE is present, it is performed on each record every time it enters the program. If other discrepancies occur in input designs, a specialized merge program is generally desirable. Attention is called to GE-225 subroutines CD225E2.001, Symbolic Tape Input/Output System and CD225C3.003, Least Key Finder. With these programming aids a merge object program may be readily assembled to accommodate unusual situations which the FORWARD merge object program will not undertake.

The programmer may also write an Output Coding Element (OCE) to process each record as its position in the final output sequence is established. Within the limitations of memory size and the number of available tape handlers, the OCE may be regarded as an independent data processing program, and can process any desired input and output files together with the merge output. In a variable merge program the Output Coding Element is not executed until the final merge level.

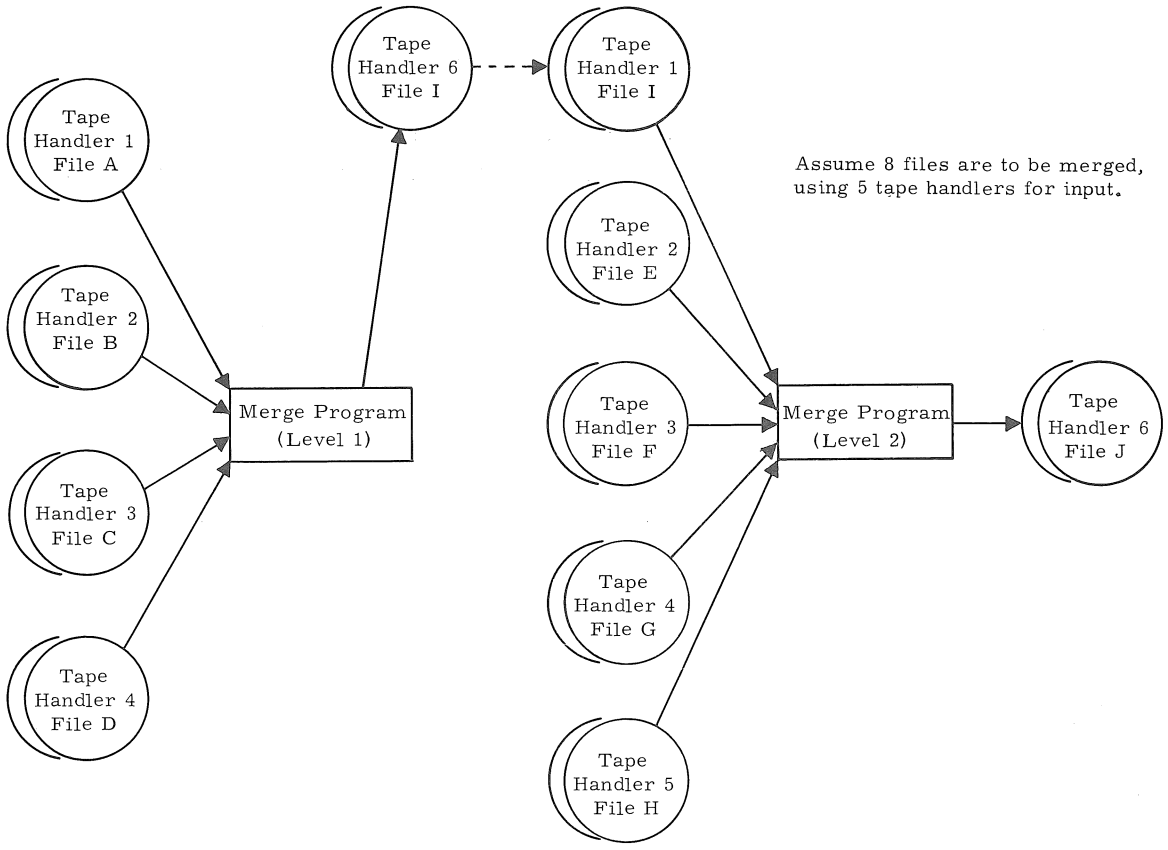


Figure 3. Flow of Variable Merge

Since block sizes are fixed and the number of files is known, the practical limit and the absolute limit for the memory area used in the merge OCE are the same, contrary to the situation in sort OCE. The merge normally assigns two buffers to each input and output file in order to achieve maximum use of the read/write/process simultaneity capability of the system. When the amount of memory is insufficient to assign two buffers for each file, the merge assigns one buffer each to one or more input files, since memory would otherwise be exceeded. A large amount of OCE can produce this condition, and decrease running efficiency, since simultaneous reading while processing is not possible on files which are assigned only one buffer. A large amount of OCE in a merge program has, however, less impact on overall program timing than in a sort program. To estimate the amount of free memory for OCE in a merge program, calculate the total buffering requirements (start with two buffers for each file) and allow 1500 words for the program plus the amount of memory needed for other User Coding Elements.

No assumption may be made about which record will be selected first when equal sequence keys are discovered. If records with duplicate keys are to be eliminated or combined while merging, such processing may be done in the merge Output Coding Element, or, it may be accomplished in a merge Squeeze Coding Element (SCE). (See Section II.)

When each intermediate output tape of a merge is closed, a checkpoint for rerun is established by writing a memory dump after the data. Thus a rerun point is established periodically throughout each merge pass. In addition, a variable merge can be discontinued at the end of any merge pass. To restart a pass, the operator merely informs the variable merge how many input files remain to be merged. It is possible to suppress the rerun memory dump with an entry of an "N" in column 15 of the output parameter card. The typewriter message then appears without the "RP." See Section X for operating instructions for the FORWARD merge object program.

IV. FORWARD PARAMETER CARDS

The FORWARD parameter cards are laid out so they can be coded for keypunching by using standard General Assembly Program coding sheets. Figure 4 shows how the card formats fit on a coding sheet. The FORWARD Parameter Card Description table (Figure 5) describes all of the fields for each of the specified parameter card types.

The SORT column of the table indicates whether the associated field may be specified for a sort program.

S	Denotes that it is permissible to specify the field but is not required.
S*	Denotes that the field must be specified.
Blank	Denotes that the field must be omitted; if the SORT column is blank for all fields in a particular card type, the card type must be omitted.

The MERGE column in the table indicates whether the associated field may be specified for a merge program.

M	Denotes that is is permissible to specify the field but is not required.
M*	Denotes that the field must be specified.
Blank	Denotes that the field must be omitted; if the MERGE column is blank for all fields in a particular card type, the card type must be omitted.

All parameter card columns which are not specified in the FORWARD Parameter Card Description table must be left blank. If desired, columns 76-80 on each card may be used for a sequence number; otherwise they must be left blank. Numeric fields should always be right column justified with leading blanks or zeros.

The right hand (ID) column in the description table is a symbolic identification of the parameter field, used to identify the field throughout the FORWARD generator and to signal parameter card errors via the high-speed printer to the programmer and the operator. (See Section VIII.) Underscored ID symbols indicate fields which must be filled out properly in order for a coherent program to be generated. When errors are detected in the fields bearing the underscored ID symbols, the FORWARD generator stops when all cards have been read and checked and does not generate an object program until the cards are corrected and resubmitted. In case an error is detected in the other fields which do not have an underscore for the ID symbols, the generator guesses at the correct value as indicated in the ACTION column, if the operator so specifies by use of console switches.

The input to the FORWARD generator consists of parameter cards and user coding elements arranged in the order in which they are presented in Figure 5 and the table which follows.

GENERAL ELECTRIC — COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL ASSEMBLY PROGRAM CODING SHEET

PROGRAMMER | PROGRAM: Sample parameter card formats | DATE | PAGE OF

	Symbol	Opr	Operand	X	REMARKS	Sequence
	1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 39 49 75	76 77 78 79 80
PROGRAM CARD (Required in sort & merge)	A A A B B Δ	C C C	D D E E P F G H	I	J J J J J J J J ΔK K K K K K K K K	
INPUT CARD (Required in sort & merge)	I N P U T Δ	A Δ B	C C C Δ D D E E	Δ	F F F F F F F F F F F F F F ΔG G G	
I-FILE CARD (Optional in sort)	I - F I L E	Δ Δ Δ	Δ Δ Δ Δ Δ Δ Δ Δ	Δ	←Cols. 31-75 for ICE file symbol list→	
ICE HEADER CARD (Optional in sort & merge)		I C E				
OUTPUT CARD (Required in sort & merge)	O U T P U T	A Δ B	C C C H D D E E	Δ	F F F F F F F F F F F F F F ΔG G G	
O-FILE CARD (Optional in sort & merge)	O - F I L E	Δ Δ Δ	Δ Δ Δ Δ Δ Δ Δ Δ	Δ	←Cols. 31-75 for OCE File symbol list→	
OCE HEADER CARD (Optional in sort & merge)		O C E				
SORT CARD (Required in sort)	S O R T Δ A	Δ B Δ	C Δ Δ G D D D D	E	F F, F F, - - - Cols. 31-53, for 3 to 8 collation tapes	
MULTICYCLE CARD	M C Y C L E	Δ A A	B B B B B B Δ Δ	Δ		
C-FILE CARD (Optional in sort)	C - F I L E	Δ Δ Δ	Δ Δ Δ Δ Δ Δ Δ Δ	Δ	A A A A A A A A A A A A A A A	
SCE HEADER CARD (Optional in sort & merge)		S C E				
KEY CARD (Optional in sort)	K E Y Δ Δ Δ	Δ Δ Δ	Δ Δ Δ Δ Δ Δ Δ Δ	Δ	A A, A A, --- Cols. 31-74, up to 15 key words	
MERGE CARD (Required in merge)	M E R G E Δ	Δ A A	B Δ Δ F C C C C	D	E E, E E, --- Cols. 31-65, for 1 to 12 input tapes	
M-FILE CARD (Optional in merge)	M - F I L E	Δ Δ Δ	A A B B Δ Δ Δ Δ	Δ	C C C C C C C C C C C C C C C	
LCE HEADER CARD (Optional in merge)		L C E				

CK-34 (1/62

Figure 4. Example of General Assembly Program Coding Format for FORWARD Parameter Cards and User Coding Elements.

FORWARD PARAMETER CARD DESCRIPTION TABLE

CARD TYPE	FIELD	COLS.	SORT	MERGE	DESCRIPTION	INTERPRETATION	ACTION IN FIRST GENERATOR PASS	ID
Program Card	AAA SRT	1-3	S*	M*	Program Type	1. Literal "SRT" specifies a sort. 2. Literal "MRG" specifies a merge. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume sort. (#1)	A1
	BB 08	4-5	S*	M*	Memory Size	1. 04 or Δ 4 specifies 4096 words. 2. 08 or Δ 8 specifies 8192 words. 3. 16 specifies 16384 words (only lower 8192 words will be used by the object-program). 4. Other: invalid.	1. Store octal 10000. 2. Store octal 20000. 3. Store octal 40000. 4. Error; assume 4096. (#1)	A2
	CCC 020	8-10	S*	M*	Record Size (words) (UNBLOCKED)	1. BCD number (not zero). 2. Other: invalid.	1. Store as a binary number. 2. Error; can't guess.	A3
	DD Δ	12-13	S	M	Program Medium	1. Blank if object-program is to be stored on cards. 2. Channel and tape (PT) number if capability of storage on tape through use of BRIDGE II is desired. 3. Other (or impossible channel and tape number): invalid.	1. Store binary zero. 2. Store as a BCD word in the form PTΔ. 3. Error; assume cards. (#1)	A4
	EE 04	14-15	S*	M*	Key Size (words)	1. BCD number (not zero) which does not exceed record size. 2. Other: invalid.	1. Store as a binary number. 2. Error; can't guess.	A5
	P N	16	S	M	API Option	1. "N" or blank if the API option is not used. 2. "Y" if the API option is to be used. 3. Others: invalid.	1. Store binary zero. 2. Store binary two. 3. Error; assume no API option (#1)	A11
	F Y	17	S	M	MOV instruction option	1. "N" or blank if object-computer does not have MOV instruction. 2. "Y" if MOV is available. 3. Others: invalid.	1. Store binary one. 2. Store binary zero. 3. Error; assume no MOV. (#1)	A6

CARD TYPE	FIELD	COLS.	SORT	MERGE	DESCRIPTION	INTERPRETATION	ACTION IN FIRST GENERATOR PASS	ID
Program Card (Cont.)	G Y	18	S	M	Single length key comparison option.	1. "Y" if key comparisons are to be made one word at a time only. 2. "N" or blank if object-program may use double length instructions whenever possible. 3. Other: invalid.	1. Store binary one. 2. Store binary two. 3. Error; assume double-length instruction permitted. (#2)	A10
	H Y	19	S	M	Three-way compare option	1. "N" or blank if object-computer does not have three-way compare instruction. 2. "Y" if three-way compare is available. 3. Other: invalid.	1. Store binary one. 2. Store binary zero. 3. Error; assume no three-way compare. (#1)	A9
	I B OR D	20	S*	M*	Tape Storage Mode	1. "B" if 20-bit binary. 2. "D" if BCD. 3. "S" if 18-bit special binary. 4. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Store binary two. 4. Error; assume BCD. (#2)	A7
	J-J	31-39	S*	M*	Program Name	1. Nine typable alphanumeric characters, not all blank. 2. All blank: invalid.	1. Store as 3 BCD words. 2. Error; assume valid. (#1)	A8
	K-K Δ	41-49	S	M	Next Program Name (For use with BRIDGE II)	1. Blank if Col. 12-13 are blank. 2. "NEX" if Col. 12-13 contain the instruction tape number but the next program is unknown. 3. Nine typable alphanumeric characters, not all blank, if the next program name is known and a tape program is being generated.	1. Store binary zero. 2. Store "NEXΔ-Δ" as 3 BCD words. 3. Store as 3 BCD words.	B8 B9 E8
Input Card	INPUT	1-5	S*	M*	Card Identification	1. Literal "INPUT". 2. Other: invalid.	1. Process next parameter. 2. Error; assume "INPUT". (#1)	B0
	A Y	8	S*	M*	Input Subroutine Option	1. "Y" indicates input subroutine is desired. It may be omitted only in sorts with ICE. 2. "N" indicates input subroutine should be omitted. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume "Y". (#1)	B1

B	10	S*	M*	ICE Option	1. "Y" indicates ICE is provided. 2. "N" indicates no ICE. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume no ICE. (#2)	B2
CCC	12-14	S	M*	Input Blocking Factor	1. BCD number (not zero); record size times blocking factor must not exceed 1000 words. 2. Blank when input subroutine is to be omitted. 3. Nonblank when input subroutine is to be omitted: invalid. 4. Condition #1 not satisfied but input subroutine is desired: invalid.	1. Store as binary number. 2. Store binary zero. 3. Error; assume blank. (#2) 4. Error; can't guess.	B3
DD	16-17	S		Sort Input Channel and Tape	1. Channel and tape (PT) number when input subroutine is desired. 2. Blank when input subroutine is not desired (always blank for merge). 3. Nonblank when input subroutine is to be omitted: invalid. 4. Condition #1 not satisfied but input subroutine is desired: invalid.	1. Store as a BCD word in the form PT△. 2. Store binary zero. 3. Error; assume blank. (#2) 4. Error; can't guess.	B4
EE	18-19	S		Alternate Input Channel and Tape	1. Blank indicates no tape swap (always blank except when input swap is wanted in sort with input subroutine generated). 2. Channel and tape number indicate swapping sort input tape. 3. Other: invalid.	1. Store binary zero. 2. Store as a BCD word in the form PT△. 3. Error; assume blank. (#1).	B5
F	31-45	S		Sort Input Label and Date	1. 15 BCD characters (Conventionally a 9-character label and a 6-character date designator) when sort input subroutine is desired. 2. Blank for merge or for sort with no input subroutine. 3. All blank but input subroutine is specified: invalid. 4. Not blank but object-program is not a sort with input subroutine: invalid.	1. Store as 5 BCD words. 2. Store binary zero. 3. Error; assume not blank. (#1) 4. Error; assume blank. (#2)	B6

CARD TYPE	FIELD	COLS.	SORT	MERGE	DESCRIPTION	INTERPRETATION	ACTION IN FIRST GENERATOR PASS	ID
Input Card (Cont.)	GGG	47-49	S		ICE Card Storage Option	1. Blank if no card storage. 2. BCD number if card storage area (#CA) is desired and ICE is present. 3. Other: invalid.	1. Store binary zero. 2. Store as a binary number. 3. Error; assume blank. (#1)	B7
I-File Card	I-FILE	1-6	S		Card Identification	1. Literal "I-FILE" and sort with ICE is specified. This Parameter Card is inserted in the input deck when the user desires to employ his own input and/or output subroutines in ICE by using the magnetic tape input/output routine of the generated program. The I-File symbols will be used by the rerun program in repositioning the tapes if a restart is necessary. 2. Not "I-FILE". 3. "I-FILE" but object-program is not to be a sort with ICE: invalid.	1. Examine col. 31-75. 2. Process as next card type. 3. Error; ignore this card.	C0
		31-75	S		ICE File Symbol List (for rerun purposes)	1. One-to six-character BCD symbols, separated by commas and terminated by blank column (up to 10 symbols). 2. Symbols containing more than six characters: invalid.	1. Store as double-length BCD words. 2. Error; truncate to six characters and proceed.	C1
ICE Header and Symbolic Cards	ICE	8-10	S	M	ICE Identification (first card)	1. Literal "ICE" is specified on Input card. 2. Not "ICE" when ICE is not specified on Input card. 3. "ICE" but ICE is not specified on Input card: invalid. 4. Not "ICE" but ICE is specified on Input card: invalid.	1. Copy ICE symbolic coding to tape until END card is reached. 2. Process as next card type. 3. Error; read and print cards until the END card, without applying them. 4. Error; can't guess.	D0
Output Card	OUTPUT	1-6	S*	M*	Card Identification	1. Literal "OUTPUT". 2. Other: invalid.	1. Process next parameter. 2. Error; assume "OUTPUT". (#1)	E0

A Y	8	S*	M*	Output Subroutine	1. "Y" indicates output is desired. It may be omitted only when OCE is present 2. "N" indicates output subroutine should be omitted. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume "Y". (#1)	E1
B N	10	S*	M*	OCE Option	1. "Y" indicates OCE is provided. 2. "N" indicates no OCE. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume "N". (#2)	E2
CCC 020	12-14	S	M	Output Blocking Factor	1. BCD number (not zero), record size times blocking factor must not exceed 1000 words. 2. Blank when output subroutine is to be omitted. 3. Nonblank when output subroutine is to be omitted: invalid. 4. Condition #1 not satisfied but output subroutine is desired: invalid.	1. Store as binary number. 2. Store binary zero. 3. Error; assume blank. (#2) 4. Error; can't guess.	E3
H N	15	S	M	RERUN Memory Dump Option	1. "Y" or blank indicates that RERUN memory dumps are to be written. 2. "N" indicates that no RERUN memory dumps are to be written. 3. Other: invalid.	1. Store binary one. 2. Store binary zero. 3. Error; assume "Y". (#1)	E9
DD 15	16-17	S	M	Output Channel and Tape	1. Channel and tape number must be specified for variable merge when output subroutine is desired; may be omitted in any sort with or without output subroutine In a sort, the tape number must be same as col. 31 and 32 of the Sort Card or left blank. 2. Blank indicates no assigned output tape. 3. Condition #1 not satisfied with a valid tape number or permissible blanks: invalid.	1. Store as a BCD word in the form PTΔ. 2. Store binary zero. 3. Error; can't guess.	E4
EE ΔΔ	18-19		M	Alternate Channel and Tape	1. Channel and tape number indicates swapping merge output tape (always blank in sort; permitted in any merge except fixed merge without output routine). 2. Blank 3. Nonblank in sort or invalid channel and tape number in merge: invalid	1. Store as a BCD word in the form PTΔ. 2. Store binary zero. 3. Error; assume blank. (#2)	E5

CARD TYPE	FIELD	COLS.	SORT	MERGE	DESCRIPTION	INTERPRETATION	ACTION IN FIRST GENERATOR PASS	ID
Output Card (Cont.)	FF	31-45	S	M	Output Label and Date	1. 15 BCD characters (conventionally a 9-character label and a 6-character date designator). Always specified for a variable merge; always specified in any sort which does not apply a C-File Card. 2. Blank when no output. 3. Blank but output routine is specified.	1. Store as 5 BCD words. 2. Process next parameters. 3. Error; assume nonblank. (#1)	E6
	GGG	47-49	S	M	OCE Card Storage Option	1. Blank if no card storage. 2. BCD number if card. 3. Other: invalid.	1. Store binary zero. 2. Store as a binary number. 3. Error; assume blank. (#1)	E7
O-File Card	O-FILE	1-6	S	M	Card Identification	1. Literal "O-FILE" and OCE is specified. This Parameter Card is inserted in the input deck when the user desires to employ his own input and/or output subroutines with OCE by using the magnetic tape input/output routine of the generated program. The O-File symbols will be used by the rerun program in repositioning the tapes if a restart is necessary. 2. Not "O-FILE". 3. "O-FILE" but object-program has no OCE: invalid.	1. Examine col. 31-75 2. Process as next card type. 3. Error; ignore this card.	F0
		31-75	S	M	OCE File Symbol List (for rerun purposes)	1. One-to six-character BCD symbols, separated by commas and terminated by blank column (up to 10 symbols). 2. Symbols containing more than six characters: invalid.	1. Store as double-length BCD words. 2. Error; truncate to six characters and proceed	F1
OCE Header and Symbolic Cards	OCE	8-10	S	M	OCE Identification (first card)	1. Literal "OCE" when OCE is specified on Output card. 2. Not "OCE" when OCE is not specified on Output card. 3. "OCE" but OCE is not specified on Output card: invalid. 4. Not "OCE" but OCE is specified on Output card.	1. Copy OCE symbolic coding to tape until END card is reached. 2. Process as next card type. 3. Error; read and print cards until the END card, without applying them. 4. Error; can't guess.	G0

Sort Card	SORT	1-4	S*		Card Identification	1. Literal "SORT" when sort is desired. 2. Not "SORT" when merge is desired. 3. Not "SORT" but sort is desired: invalid. 4. "SORT" but merge is desired: invalid.	1. Process next parameter. 2. Process as next card type. 3. Error; assume "SORT". (#1) 4. Error; ignore this card and go on to next card type.	H0
	A Y OR Δ	6	S		Collation Label Option LABEL	1. "Y" if a C-File card follows, specifying the label to be used on collation tapes (otherwise output label is used). 2. "N" or blank indicates no C-File card follows. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume "N". (#2)	H1
	B Δ OR T	9	S		Scattered Key Option	1. Blank if standard key (in order of decreasing significance, starting in the first word of the record). 2. "T" indicates a scattered key (described in Key card), and the record format is not to be altered to "standard" key on output. 3. "P" indicates a scattered key (described in Key card), and the record format is to be altered as follows: Key is moved to the front in "standard" form and the remaining words of the record are moved compactly in their original order to follow the key (record size is unchanged). 4. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Store binary two. 4. Error; assume standard key. (#1)	H2
	C N	12	S		Single-Pass Sort Option SELDOM USED BUT IF USING USE WITH CAUTION	1. "Y" indicates that good input sequence is expected and the presort may ignore the rotary-merge if it produces a single output string. The sort will use the output blocking factor on its collation tapes when this option is applied. This option is ignored when a "T" type scattered Key option is used. (See column 9 entry, above.) 2. "N" or blank indicates sort is to determine optimum collation tape blocking and always execute rotary-merge. The best choice is "N" unless specific knowledge predicts a one-pass sort. Should be "N" if OCE is used. 3. Other: invalid.	1. Store binary one. 2. Store binary zero. 3. Error; assume "N". (#2)	H11

GOOD IDEA TO USE COLLATION TAPE LABEL CAN SEE ON TYPEWRITER ITEM FOR SORT ITEMS DONE

Δ - remove C-FILE CARD

CARD TYPE	FIELD	COLS.	SORT	MERGE	DESCRIPTION	INTERPRETATION	ACTION IN FIRST GENERATOR PASS	ID
Sort Card (Cont.)	G	15	S		Multicycle Sort Option.	1. "N" or blank if this option is to be omitted. 2. "Y" if this option is to be used and an "MCYCLE" parameter card follows. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Assume "N" or blank (#2)	H15
	D-D	16-19	S		Communication Storage	1. BCD number in the form NNNN or ΔNNN indicates the number of words of upper memory (lower bank if 16k) to be reserved for Communication Storage (#CS). The number must not exceed 1999 in 4k memories or 3999 in 8k or 16k memories. 2. Blank indicates no communication storage. 3. Other: invalid.	1. Store as a binary number. 2. Store binary zero. 3. Error; assume blank. (#2)	H4
	E	20	S		SCE Option	1. "Y" indicates SCE is provided. 2. "N" or blank indicates no SCE. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume blank. (#2)	H5
	FF,...	31-53	S*		Collation Tape List	1. Channel and handler numbers of collation tapes, separated by commas (e.g., 12, 13, 14, ...). From three to eight collation tapes may be listed; if a specific output handler is assigned, it must be the first in the list (see Output card). Blank column terminates the list. Channel and handler specified in input card may be used here if last in group. 2. Other (or duplicates, impossible numbers, etc.): invalid.	1. Store each channel and tape as a BCD word in the form PT. 2. Error; can't guess.	H6
Multi-cycle Card	MCYCLE	1-6	S		Card Identification	1. Literal "MCYCLE" when multicycle sort option is specified in the sort card. 2. Not "MCYCLE" if program is not a sort with a multicycle option. 3. Not "MCYCLE" but multicycle option is specified in the sort card: invalid. 4. "MCYCLE" but multicycle option is not specified in sort card or program is not a sort: invalid.	1. Process next. 2. Process as next card type. 3. Error; assume multicycle option is not wanted and process this as next card. 4. Error; ignore this card and go on to next card.	P1

	AA	9-10	S		Channel and tape no. to be used as storage for the pre-sort sub-program.	1. Controller channel and tape handler number. 2. Other (impossible numbers, or the same as other working tapes): invalid.	1. Store as a BCD number in the form PTΔ. 2. Error; can't guess.	P2
	B-B	12-17	S		Number of records	1. BCD number indicating the number of records to be sorted in each cycle of a multicycle sort. 2. Other: invalid.	1. Store as binary number. 2. Error; can't guess.	P3
C-File Card	C-FILE	1-6	S		Card Identification	1. Literal "C-FILE" if collation label option is specified on Sort card. (This option is used when a unique label is desired for collation tapes. Otherwise the sort writes the output label on the collation tapes.) 2. Not "C-FILE" when collation label option is not specified. 3. "C-FILE" but collation label option is not specified, or program is not a sort: invalid. 4. Not "C-FILE" but collation label option is specified on Sort card: invalid.	1. Process next field. 2. Process as next card type. 3. Error; ignore this card and go on to next card type. 4. Error; assume collation label is not wanted and process as next card type.	I0
	AA	31-45	S		Collation Tape Label and Date	1. Nonblank 2. All blank: invalid.	1. Store as 5 BCD words. 2. Error; apply anyway. (#1)	I1
SCE Header and Symbolic Cards	SCE	8-10	S	M	SCE Identification (first card)	1. Literal "SCE" when SCE is desired. 2. Not "SCE". 3. "SCE" in a sort when it has not been specified in the Sort card: invalid. 4. Not "SCE" but it has been specified in the Sort card: invalid.	1. Copy SCE symbolic coding to tape until END card is reached. 2. Process as next card type. 3. Error; assume SCE is desired. (#1) 4. Error; assume no SCE and process as next card type. (#2)	J0
Key Card	KEY	1-3	S	M	Card Identification	1. Literal "KEY" when the Scattered Key Option is selected in the Sort or Merge card. 2. Not "KEY" if the Scattered Key Option is not selected in the Sort or Merge Card.	1. Examine col. 31-75. 2. Process as next card type, i.e., as end of deck)	M0

CARD TYPE	FIELD	COLS.	SORT	MERGE	DESCRIPTION	INTERPRETATION	ACTION IN FIRST GENERATOR PASS	ID
Key Card (Cont.)						3. Not "KEY" but Scattered Key Option is selected: invalid. 4. "KEY" but Scattered Key Option is not selected: invalid.	3. Error; assume Standard Key and conclude first generator pass. 4. Error; assume Standard Key and conclude first generator pass.	
	AA,....	31-74	S	M	Key Description FIRST WORD 00 44 00,04,08,00	1. Key word positions in order of decreasing significance. Up to 15 key words may be specified, anywhere in the first thirty (00 - 29) words of the record. Each key word position is indicated as a two-digit BCD number; they are separated by commas and the generator expects exactly as many words as the "key size" specified in the Program card. 2. Condition #1 not satisfied: invalid.	1. Store as binary numbers. 2. Error; can't guess.	M1
Merge Card	MERGE	1-5		M*	Card Identification	1. "MERGE" when merge object-program is desired. 2. Other: invalid.	1. Process next parameter. 2. Error; assume "MERGE". (#1)	K1
	AA	9-10		M*	Merge Type	1. Blank indicates variable merge. 2. BCD number (not zero but no greater than twelve) specifies a fixed merge with this number of input files. 3. Other: invalid.	1. Store binary zero. 2. Store as a binary number. 3. Error; assume variable merge. (#1)	K2
	B	12		M	LCE Option	1. "Y" indicates LCE is provided. (Interpreted in variable merge only.) LCE is used to generate input labels when label check is desired on all inputs. 2. "N" or blank if LCE is not provided 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume no LCE. (#2)	K3
	F	15		M	Scattered Key Option	1. "N" or blank indicates no scattered key is desired. 2. "Y" indicates Scattered Key Option is desired. This option does not alter the record format in the merge. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume "N". (#1)	H2

	C-C	16-19		M	Communication Storage	1. BCD number in the form NNNN or ΔNNN indicates the number of words of upper memory (lower bank if 16k) to be reserved for Communication Storage (#CS). The number must not exceed 1999 in 4k memories or 3999 in 8k or 16k memories. 2. Blank indicates no communication storage. 3. Other: invalid.	1. Store as a binary number. 2. Store binary zero. 3. Error; assume blank. (#2)	H4
	D	20		M	SCE Option	1. "Y" indicates SCE has been provided. 2. "N" or blank indicates no SCE. 3. Other: invalid.	1. Store binary zero. 2. Store binary one. 3. Error; assume blank. (#2)	H5
	EE,...	31-65		M	Variable Merge Inputs	1. Blank if fixed merge. 2. Channels and handlers of variable merge input tapes, separated by commas (e.g., 12, 13, 14,...). From one to twelve input tapes may be listed. Blank column terminates the list. 3. Condition #2 not satisfied in a variable merge: invalid.	1. Store binary zero. 2. Store as BCD words in the form PTΔ. 3. Error; can't guess.	K4
M-File (one for each fixed merge input)	M-FILE	1-6		M	Card Identification	1. Literal "M-FILE" if fixed merge. 2. Not "M-FILE" but not fixed merge: invalid. 3. "M-FILE" but not fixed merge: invalid. 4. Not "M-FILE" but program is fixed merge: invalid. 5. More "M-FILE" cards than the number of inputs specified on the Merge card in cols. 9-10: invalid. 6. Fewer "M-FILE" cards than specified on the Merge card: invalid.	1. Examine next parameter. 2. Process as next card type. 3. Error; ignore this and any other M-File cards. 4. Error; can't guess. 5. Error; ignore extra M-File cards. 6. Error; can't guess.	L0
	AA	12-13		M	Input Channel and Tape	1. Legitimate channel and tape number which is different from any other mentioned in another M-File card (or in col. 14-15). 2. Other: invalid.	1. Store as a BCD word in the form PTΔ. 2. Error; can't guess.	L1

CARD TYPE	FIELD	COLS.	SORT	MERGE	DESCRIPTION	INTERPRETATION	ACTION IN FIRST GENERATOR PASS	ID
M-File (Cont.)	BB	14-15		M	Alternate Channel and Tape	1. Blank if no tape swap for this file. 2. Channel and tape number of alternate tape (see condition #1 of Input channel and tape.)	1. Store binary zero. 2. Store as a BCD word in the form PTΔ.	L2
	C-C	31-45		M*	Label and Date Designator	1. 15 BCD characters (conventionally a 9-character date designator). 2. Blank.	1. Store as 5 BCD words. 2. Error; assume valid. (#1)	L3
LCE Header and Symbolic Cards	LCE	8-10		M	LCE Identification (first card)	1. Literal "LCE" when specified in Merge card. 2. Not "LCE" when not specified in Merge card. 3. "LCE" but Merge card does not specify LCE: invalid. 4. Not "LCE" but Merge card specifies LCE: invalid.	1. Copy LCE symbolic coding to tape until END card is reached. 2. Conclude first Generator pass. 3. Error; ignore LCE. 4. Error; assume no LCE is desired and conclude first Generator pass.	N1

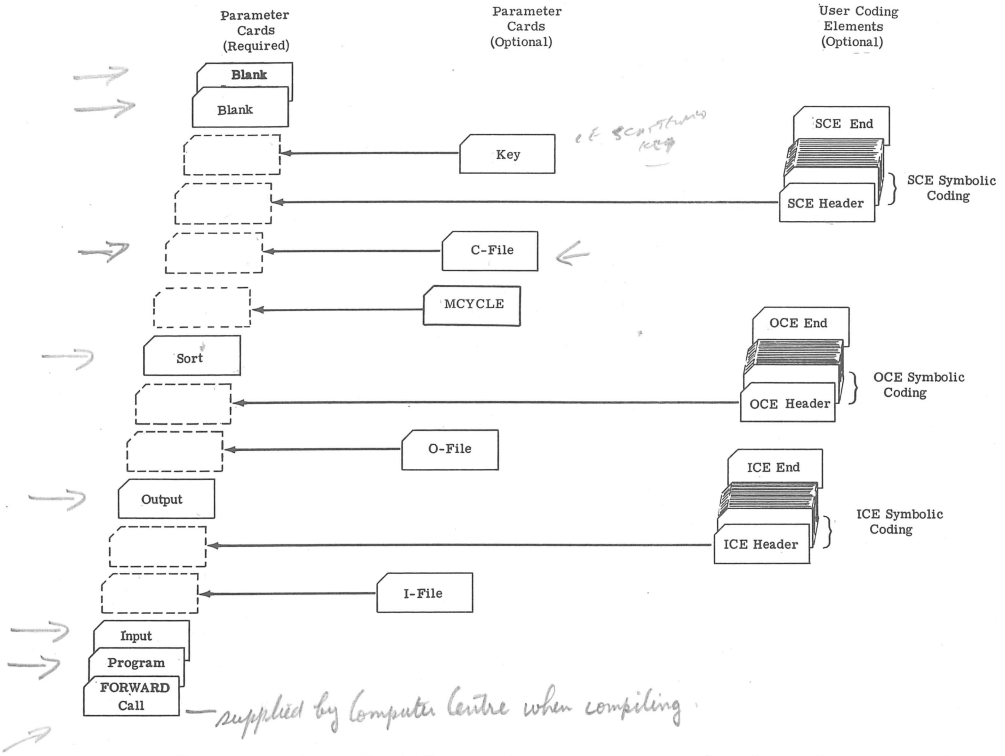


Figure 5. FORWARD Sort Generator Parameter Cards and User Coding Element Cards - Input Arrangement

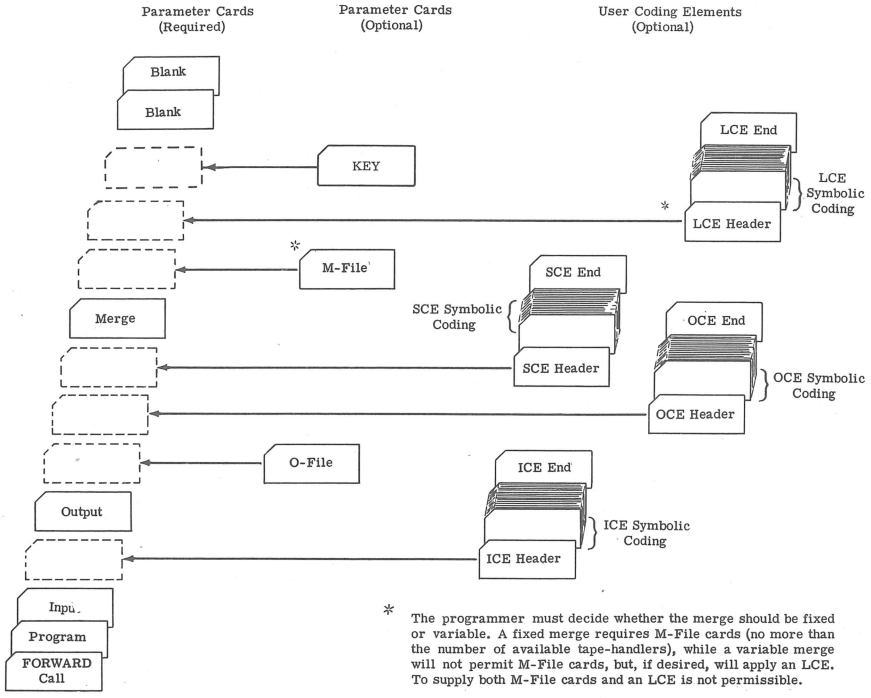


Figure 6. FORWARD Merge Generator Parameter Cards and User Coding Element Cards - Input Arrangement

V. FORWARD USER CODING ELEMENTS

User Coding Elements are written in the symbolic coding of the General Assembly Program. When common information is needed by various User Coding Elements (UCE) in a sort program it must be stored in the communication storage memory area, having an address symbol of #CS. If two or more User Coding Elements are written for a sort, they must have no cross-referencing or duplication of symbols, because they may or may not appear in memory at the same time. Therefore, no assumption may be made. Communication storage is also available in a merge program but all merge User Coding Elements are contained in memory at the same time, so that cross referencing of User Coding Elements is permissible. The FORWARD generator will not detect erroneous cross referencing or symbol duplication. User Coding Elements should not use symbols starting with #, except for the special symbols noted in the following pages.

An identifying header card precedes each User Coding Element. Columns 8-10 (the "Operation" portion) contains a three-letter abbreviation indicating the type of User Coding Element. An ordinary General Assembly Program "END" card terminates the User Coding Element, although it is not applied during the program assembly.

Because the operand portion of a punched card instruction must be a multiple of 128, and document handler instructions must be a multiple of 64, a special memory area for card and document storage (Card Storage) is reserved in lower memory for Input Coding Element (ICE) or Output Coding Element (OCE) when specified by the user. This area will begin with symbolic location #CA and extend to accommodate the number of words specified by the programmer. It may be specified separately for ICE and OCE. #CA is assigned an address which is a multiple of 128.

The following coding may be used in special cases:

GENERAL ELECTRIC — 225 GENERAL ASSEMBLY PROGRAM CODING SHEET

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

PROGRAMMER | PROGRAM

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31
B A C K	E Q U	*		
	O R G	X X X X		} ORG #CS, for example
				} desired data or coding for #CS area
	O R G	B A C K		} the remaining program is assembled with previous part

Normally, User Coding Elements should not contain the pseudo-instruction ORG to designate the origin or starting location in memory. However, buffers and working storage areas may be reserved by applying the pseudo-instruction, BSS, to reserve a block of memory locations of a specified amount. Arbitrary use of ORG is almost certain to lead to difficulty. If, however, it is desired to generate the initial contents of the communication storage area (#CS), the special case coding may be used as was just illustrated.

When the Scattered Key Option is used in a sort program containing User Coding Elements, the following conditions prevail:

The record presented to or received from ICE is in the input format.

The record is reformatted to "standard" form by the sort for all of its intermediate processes. Thus SCE always receives the data in "standard" format. (Standard record format is described as having key words at the beginning of the record in descending significance with the remaining data following compactly.)

Restoration to the input format is automatic when the "T" (Temporary) form of the Scattered Key Option is chosen. The restoration is made before the record is presented to OCE.

No automatic restoration of input format is provided when the "P" (Permanent) form of the Scattered Key Option is chosen. OCE then, of course, receives the records in "standard" form.

Example: Assume the key is in words 2, 6, 5, 7, and 8 of a ten word record. (Numbers represent position of words in the input record.)

Input format (processed by ICE and, if "T" option, by OCE)	"Standard" format (processed by SCE and, if "P" option, by OCE)	
0	2	} "standard" key
1	6	
2	5	
3	7	
4	8	
5	0	
6	1	
7	3	
8	4	
9	9	

The Merge Scattered Key Option extension permits execution of merge object programs using data that have been sorted by using the Sort Scattered Key Option. Thus the need for extensive use of User Coding Elements to handle the merging of records sorted by use of scattered keys has been eliminated. The Scattered Key Option is for merging files with a record key which does not conform to normal assumptions, i.e., where the key words are together at the beginning of the record. The user can specify the use of the Merge Scattered Key Option by entering a "Y" in column 15 of the merge parameter card and including a key parameter card to describe the

merge scattered key. The merge object program will contain the necessary procedures based on the key card information to handle the scattered key. Although the Squeeze Coding Element (SCE) must accept a rearranged record format in sort object programs using the Scattered Key Option, merge object programs using the Scattered Key Option do not involve any rearrangement of record format and therefore SCE in merge object programs processes records in the same format as they appear in the input. Unless a permanent change in the record format is desired for subsequent programs, the temporary (T) form of the Sort Scattered Key Option should be selected in the sort card.

When a merge object program using the Scattered Key Option follows a sort object program "T" is always selected for column 12 of the sort parameter card, a "Y" is entered in column 15 in the merge card, and the key cards prepared for both the sort and the merge object programs are identical. The result is that the input format is used for the sort object program output and throughout the merge object program, including the output.

Care must be taken to assure that User Coding Elements do not exceed memory size. An estimate of memory requirements may be made by considering the following factors:

About 1550 words are required for the FORWARD program coding.

Input Coding Element (ICE) and Output Coding Element (OCE) never appear in memory at the same time in a sort program.

The Squeeze Coding Element (SCE) appears in each part of a sort program in which it is used. The Label Coding Element (LCE), Input Coding Element, and Squeeze Coding Element appear in each part of a merge program in which they are used.

The FORWARD system does not use the upper half of a 16,384-word memory. This area may be used by User Coding Elements or may be used while sorting by a separate "priority interrupt" program.

The sort blocking factor will be the largest integer not exceeding:

$$\frac{\text{memory size -1550 -OCE -SCE -\#CA area -\#CS area}}{\text{2 x record size x number of collation tapes}}$$

The presort subprogram requires storage area for two input buffers and two output buffers. Both the merge program and the rotary-merge subprogram of a sort program require storage area for two input buffers for each input to a given merge level and another storage area for two output buffers.

If extra index groups are present, both the generated sort and merge object programs assume that group 0 is set whenever control is given to them. User Coding Elements may use the remaining groups, except for group 1 word 0 (absolute word 4). Neither the sort nor the merge object program requires User Coding Elements to preserve index register contents nor does either program preserve index register contents for User Coding Elements.

SORT CODING ELEMENTS

(See Section IV for organization of sort source cards.)

Input Coding Element (ICE)

ICE is written when preprocessing of the data record to be sorted is desired. (See Section II.)

The first instruction of the ICE should be a BRU to starting functions, if any. The second instruction should be a BRU to ending functions, if any. Both of these instructions may be a BRU 1,1 when starting or ending functions are not used. The third instruction is the normal entry. All calls to the Input Coding Element are SPB instructions specifying modification word 1. The sort program is always re-entered with the instruction BRU 1,1.

ENTRY/EXIT CONDITIONS

No information is provided to the ICE starting or ending functions by the sort.

The normal entry is called with the address of the input record in the A-register.

When control is returned to the sort, ICE should exit either with the output record address in the A-register (when the record is to be sorted) or with zero in A (when the record is to be ignored).

SPECIAL SORT SYMBOLS

SPB #SR ,1	Sort advances a new input record without sorting the previous record, then returns control to the instruction following the SPB.
SPB #SW ,1	The sort processes the record whose address is contained in the A-register without advancing a new input record, then returns control to the instruction following the SPB.
BRU #SE	Transfers control to the sort end-of-input functions, one of which is to execute the ICE ending functions.

The starting entry is called only once, before any input has been read. The ending entry is called only once, when all data coming into the sort has been exhausted. The sort still accepts data records from the Input Coding Element. The normal entry is called each time control is returned to the sort and a new record is produced by the sort input routine.

CONVENTIONS

Instruction SPB #SR ,1 must not be executed during ICE ending functions.

Records should not be given to the sort during the ICE starting function.

If no sort input subroutine is generated, the input functions must be performed by ICE.

If the record size is an even number, each record given to the sort must begin in an even numbered location. When the sort input subroutine is used for even numbered record sizes, each record given to ICE will begin in an even numbered location.

Header: A literal "ICE" is entered in columns 8-10.

EXAMPLES:

Figure 7 shows the data flow using ICE where a report is to be prepared about blonde female employees, using the payroll master file for input data.

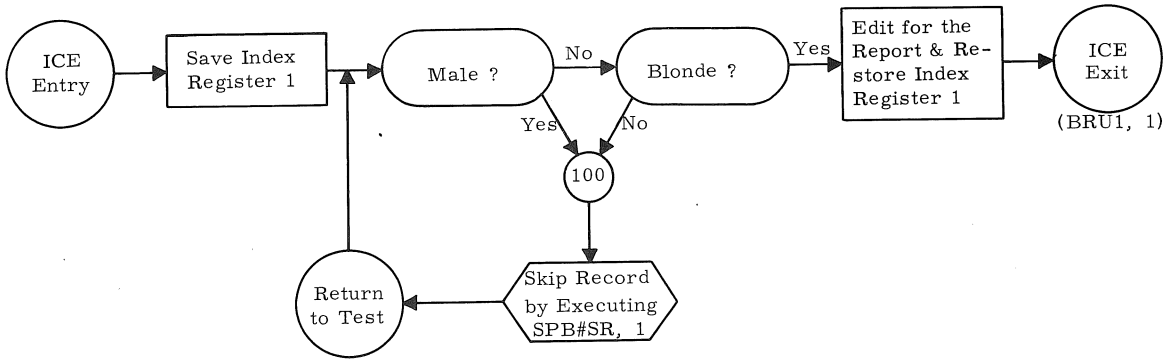
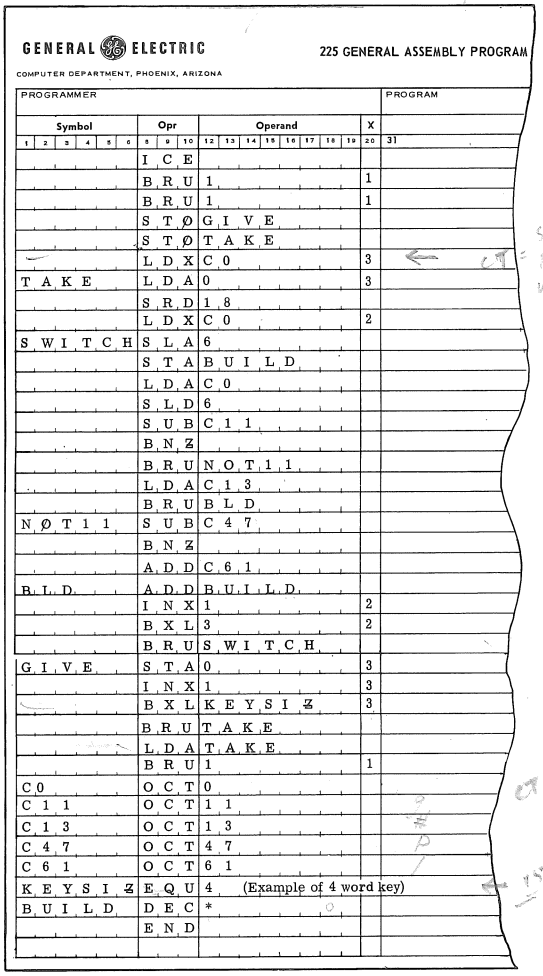


Figure 7. Example of Flow of ICE

Figure 8 shows the General Assembly Program symbolic coding for ICE to alter the key so that a blank " Δ " will sort as a least value BCD character. This routine changes all of the characters in a BCD key so that " Δ " (octal 60) effectively becomes the smallest for sorting purposes. It assumes all words converted are standard BCD data containing neither "#" nor octal 77 as characters. OCE restores the original key for subsequent processing. See Figure 11 for an example of coding for OCE to restore the original key.

Figure 9, shows the General Assembly Program symbolic coding for ICE to alter the key so that letters sort smaller than numbers. This routine changes all of the characters in a BCD key so that " Δ " (octal 60) effectively becomes the smallest for sorting purposes, followed by alphabetics and then numerics. No special arrangements are made for special characters (*, $, #, /, %, etc.). The routine assumes that all converted words are standard BCD data not containing "+" (octal 20) as a character. OCE restores the original key for subsequent processing. See Figure 12 for an example of coding for OCE to restore the original key.



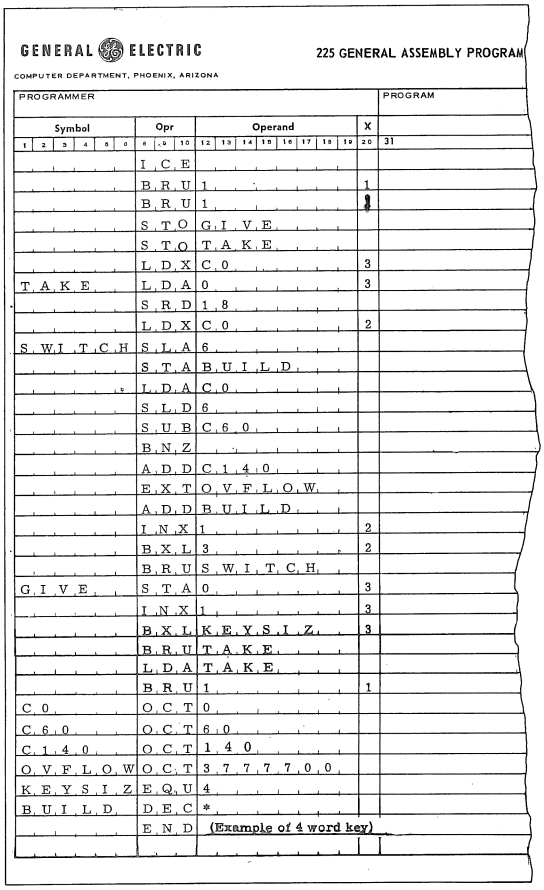
GENERAL ELECTRIC — 225 GENERAL ASSEMBLY PROGRAM

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

PROGRAMMER — PROGRAM

Symbol	Opr	Operand	X
	ICE		
	BRU	1	1
	BRU	1	1
	STØ	GIVE	
	STØ	TAKE	
	LDX	C0	3
TAKE	LDA	0	3
	SRD	18	
	LDX	C0	2
SWITCH	SLA	6	
	STA	BUILD	
	LDA	C0	
	SLD	6	
	SUB	C11	
	BNZ		
	BRU	NOT11	
	LDA	C13	
	BRU	BLD	
NØT11	SUB	C47	
	BNZ		
	ADD	C61	
BLD	ADD	BUILD	
	INX	1	2
	BXL	3	2
	BRU	SWITCH	
GIVE	STA	0	3
	INX	1	3
	BXL	KEYSIZ	3
	BRU	TAKE	
	LDA	TAKE	
	BRU	1	1
C0	OCT	0	
C11	OCT	11	
C13	OCT	13	
C47	OCT	47	
C61	OCT	61	
KEYSIZ	EQU	4 (Example of 4 word key)	
BUILD	DEC	*	
	END		

Figure 8. Coding for ICE to Alter the Sequence Key so Blanks "△" Sort as Least BCD Character Value



GENERAL ELECTRIC — 225 GENERAL ASSEMBLY PROGRAM

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

PROGRAMMER — PROGRAM

Symbol	Opr	Operand	X
	ICE		
	BRU	1	1
	BRU	1	1
	STO	GIVE	
	STO	TAKE	
	LDX	C0	3
TAKE	LDA	0	3
	SRD	18	
	LDX	C0	2
SWITCH	SLA	6	
	STA	BUILD	
	LDA	C0	
	SLD	6	
	SUB	C60	
	BNZ		
	ADD	C140	
	EXT	OVFLOW	
	ADD	BUILD	
	INX	1	2
	BXL	3	2
	BRU	SWITCH	
GIVE	STA	0	3
	INX	1	3
	BXL	KEYSIZ	3
	BRU	TAKE	
	LDA	TAKE	
	BRU	1	1
C0	OCT	0	
C60	OCT	60	
C140	OCT	140	
OVFLOW	OCT	377777000	
KEYSIZ	EQU	4	
BUILD	DEC	*	
	END	(Example of 4 word key)	

Figure 9. Coding for ICE to Alter the Sequence Key so Letters Sort Smaller than Numbers

Output Coding Element (OCE)

The Output Coding Element is written if postprocessing is to be executed when the position of each record in the final merge output sequence is established.

In the symbolic coding of OCE the first instruction should be a BRU to starting functions, if any. The second instruction should be a BRU to ending functions, if any. These two instructions

must be BRU 1, 1 when starting or ending functions are not used. The third instruction is the normal entry. All calls to the OCE will be with SPB instructions specifying modification word 1. The sort is always re-entered with the instruction BRU 1, 1.

ENTRY/EXIT CONDITIONS

No information is provided to the starting or ending functions by the sort program.

The normal entry is called with the address of the input record in the A-register.

When control is returned to the sort, OCE should exit either with the output record address in the A-register (when the record is to be written) or with zero in A (when the record is to be ignored).

SPECIAL SORT SYMBOLS

SPB #MR,1	Sort produces a new record without writing the previous record, then returns control to the instruction following the SPB.
SPB #MW,1	Causes the sort to write the record whose address is contained in the A-register, then returns control to the instruction following the SPB.
#POP	Symbolic origin of the symbolic input/output subroutine file parameter list for the collation tape output routine. If no sort output routine is generated, OCE may obtain a free tape unit and buffers from words #POP+7 through #POP+12. This tape unit may be rewinding at the beginning of OCE.

The starting entry is called only once, before any input has been read. If desired, it may write records on the sort output. The ending entry is called only once, at the time when all data coming into the sort has been exhausted. The sort output file is still open. The normal entry is called each time control is returned to the sort and after a new record is produced by the sort.

CONVENTIONS

The instruction SPB #MR,1 must not be executed during the OCE starting and ending functions.

If no sort output subroutine is generated, the output functions must be performed by OCE. In this case, the instruction SPB #MW,1 must not be executed and, during normal processing, BRU 1, 1 advances a sorted input record only.

If record size is even numbered, each record given to OCE begins in an even numbered location. If the sort output subroutine is used for even numbered record sizes, each output record must begin in an even numbered location.

Header: A literal "OCE" is entered in columns 8-10.

EXAMPLES:

Figure 10 shows the data flow where the output of a sort (or merge) is to be printed as a report, and is not to be written on tape at all.

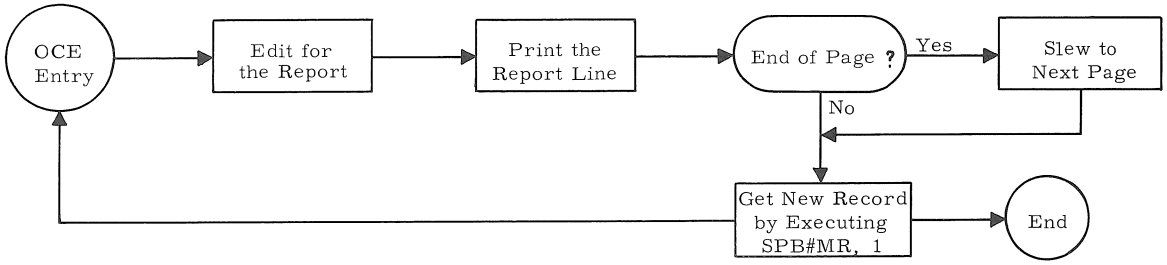


Figure 10. Example of Flow of OCE

Figure 11 shows the General Assembly Program symbolic coding for OCE to restore the key originally altered by ICE (Figure 8) so that blanks (Δ) would sort as the least BCD character value.

Figure 12 shows the General Assembly Program symbolic coding for OCE to restore the key originally altered by ICE (Figure 9) so that letters would sort smaller than numbers.

GENERAL ELECTRIC — 225 GENERAL ASSEMBLY PROGRAM

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

PROGRAMMER — PROGRAM

Symbol	Opr	Operand	X
	ØCE		
	BRU 1		1
	BRU 1		1
	STØ SINK		
	STØ SWIM		
	LDX CN0		3
SINK	LDA 0		3
	SRD 18		
	LDX CN0		2
ALTER	SLA 6		
	STA RESUME		
	LDA CN0		
	SLD 6		
	BNZ		
	BRU NØBLK		
	ADD CN60		
	BRU ACC		
NØBLK	SUB CN13		
	BNZ		
	BRU NØT13		
	LDA CN11		
	BRU ACC		
NOT13	ADD CN12		
ACC	ADD RESUME		
	INX 1		2
	BXL 3		2
	BRU ALTER		
SWIM	STA 0		3
	INX 1		3
	BXL KEYWDS		3
	BRU SINK		
	LDA SINK		
	BRU 1		1
CN0	ØCT 0		
CN11	ØCT 11		
CN12	ØCT 12		
CN13	ØCT 13		
CN60	ØCT 60		
KEYWDS	EQU 4	(Example: 4 word key)	
RESUME	DEC *		
	END		

Figure 11. Coding for OCE to Restore the Sequence Key Altered by ICE to Sort "Δ" as Least BCD Character Value

GENERAL ELECTRIC — 225 GENERAL ASSEMBLY PROGRAM C

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

PROGRAMMER — PROGRAM

Symbol	Opr	Operand	X
	OCE		
	BRU 1		1
	BRU 1		1
	STO SINK		
	STO SWIM		
	LDX C0*		3
SINK	LDA 0		3
	SRD 18		
	LDX C0*		2
ALTER	SLA 6		
	STA RESUME		
	LDA C0*		
	SLD 6		
	BZE		
	ADD C40*		
	ADD C20*		
	EXTOVFLO*		
	ADD RESUME		
	INX 1		2
	BXL 3		2
	BRU ALTER		
SWIM	STA 0		3
	INX 1		3
	BXL KEYWDS		3
	BRU SINK		
	LDA SINK		
	BRU 1		1
C0*	OCT 0		
C40*	OCT 40		
C20*	OCT 20		
OVFLO*	OCT 377777000		
KEYWDS	EQU 4	(Example: 4 word key)	
RESUME	DEC *		
	END		

Figure 12. Coding for OCE to Restore the Sequence Key Altered by ICE to Sort Letters Smaller than Numbers

Squeeze Coding Element (SCE)

The Squeeze Coding Element is used if very many records with duplicate keys exist within the data being sorted so that the extra keys may be eliminated by combining them or by simply ignoring them.

The sort calls for the first instruction of the Squeeze Coding Element with an SPB instruction specifying modification word 1. The sort is re-entered with the instruction BRU 1, 1.

ENTRY/EXIT CONDITIONS

The addresses of two records having the same key are in the A- and Q-registers, respectively, when the SCE is entered.

If the A-register record is to be kept, SCE will exit with the original contents of A preserved.

If the Q-register record is to be kept, SCE will exit with the original contents of Q preserved.

If either record is deleted, the contents of the appropriate register should be made zero. If both registers are made zero, the Q-register record is kept.

SCE is executed whenever two records with identical sequence keys are encountered in the sort.

CONVENTIONS

Deleting both records in SCE is not permitted although both may be kept if necessary. If SCE exits with zero in the A-register, the Q-register record will be kept.

No assumption may be made about the order in which duplicate records will be encountered, and any choices should be based upon information contained in the data records.

Sequence keys are expected by the sort to take one or more full words. If non-key data is contained in the least significant key word, duplicate sequence keys may not be recognized.

Whenever both the key and the record contain an even number of whole words, the origins of all records processed by SCE begin in even memory locations. Otherwise, the record origins might be even or odd, depending upon optimization considerations for the move and comparison routines in the object program. For this reason, SCE may assume even origin whenever both key and record sizes are even, but otherwise it must be programmed to accept a mixture of even and odd record origins.

The Squeeze Coding Element assumes the origin of all records to be even whenever both the key and record contain an even number of whole words.

All processing of data records should be performed within the record storage areas; A- and Q-registers will not be examined to detect new record origins.

Header: A literal "SCE" is entered in columns 8-10.

EXAMPLES:

Figure 13 shows the flow of data in a presort program using SCE to skip records with duplicate keys or to keep records with duplicate keys.

Figure 14 presents an example of the use of General Assembly Program coding for SCE. For the case problem in the example, assume that word "ten" of each record contains a count of the number of suppressed records which have the same key as the record being processed. Word "eleven" contains an amount accumulation for all of the records. The SCE routine summarizes the counts and amounts of the Q-register record and eliminates the A-register record.

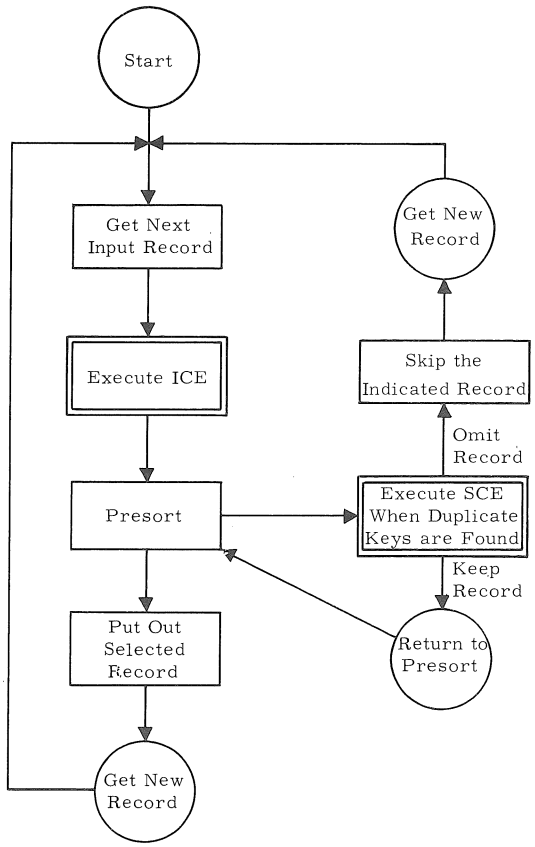


Figure 13. Flow of Presort with ICE and SCE

GENERAL ELECTRIC 225 GENERAL

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

PROGRAMMER PR

Symbol	Opr	Operand	X
	SCE		
	DST	2	
	LDA	10	3
	ADO		
	ADD	10	2
	STA	10	3
	LDA	11	3
	ADD	11	2
	STA	11	3
	LDZ		
	BRU	1	1
	END		

Figure 14. Coding for SCE to Summarize and Delete Records with Duplicate Sequence Keys

Data Flow Using Sort Coding Elements

Figure 15 illustrates the flow of data for the presort subprogram portion of a sort program.

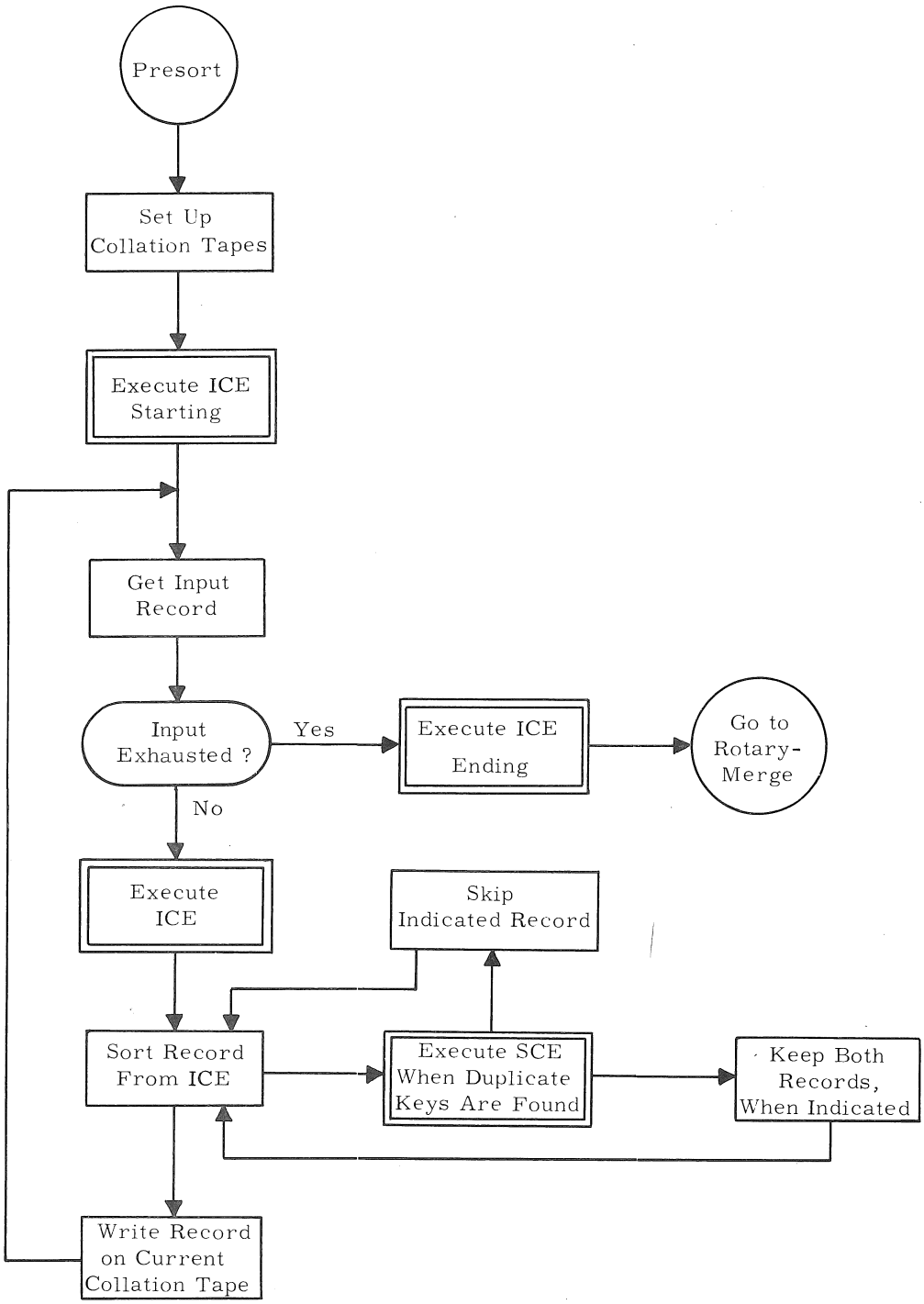


Figure 15. Flow of Sort - Presort Subprogram with ICE and SCE

Figure 16 shows the data flow during the rotary-merge subprogram portion of a sort program.

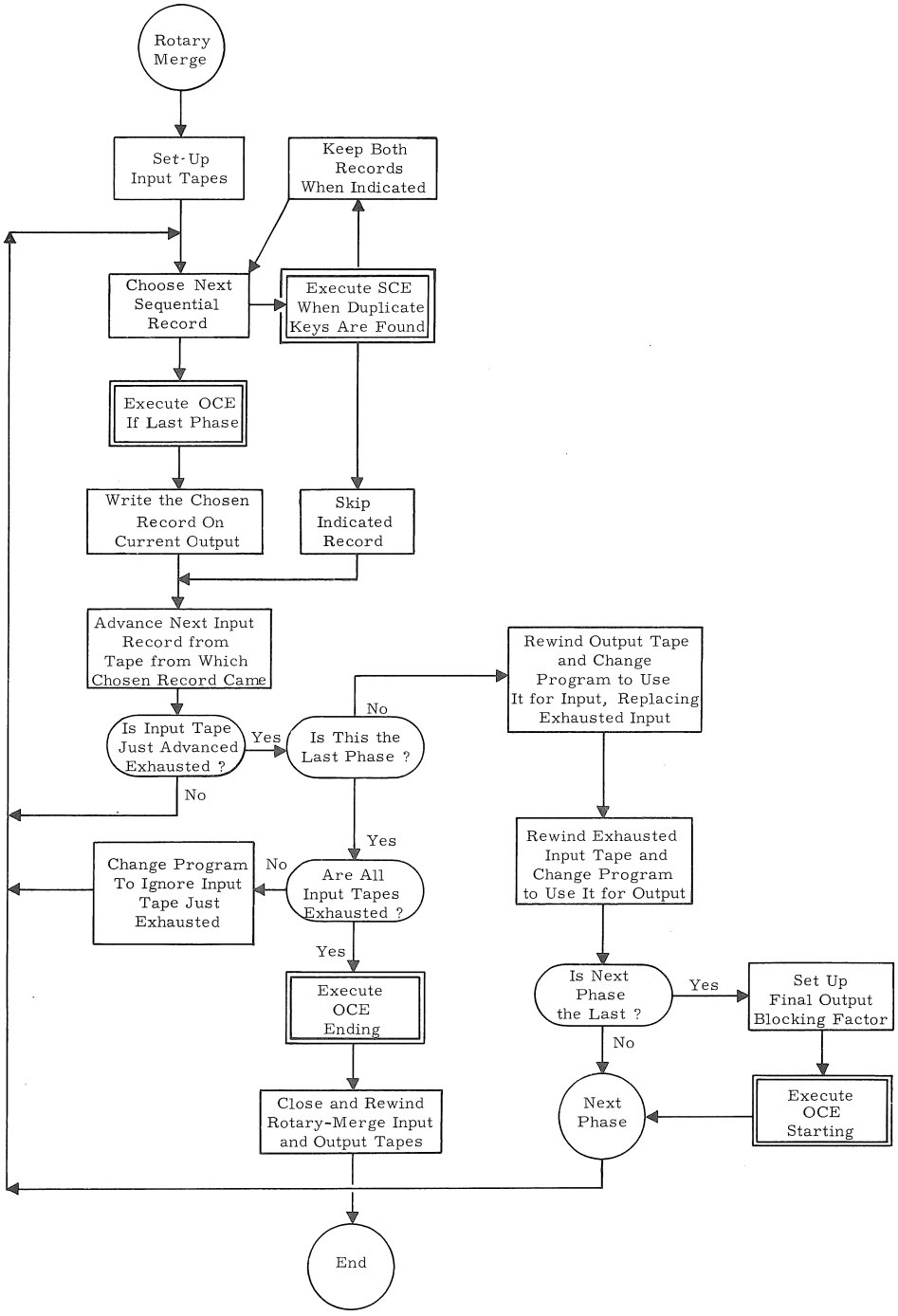


Figure 16. Flow of Sort - Rotary-Merge Subprogram with SCE and OCE

MERGE CODING ELEMENTS

(See Section IV for organization of merge source cards.)

Input Coding Element (ICE)

The Input Coding Element is prepared if record sizes and block sizes of all inputs are the same and are constant, but preprocessing is desired on some or all files; this would usually be to standardize the keys.

The merge calls the first instruction of the input coding element with an SPB instruction specifying modification word 1. The merge is re-entered with the instruction BRU 1, 1.

ENTRY/EXIT CONDITIONS

The address of the input is in A-register when ICE is called.

Each input record is submitted to ICE as soon as it is advanced into input record storage, before the merge processes it.

CONVENTIONS

All preprocessing must be done within the input storage area, the A-register will not be examined to detect a new record origin.

If various files or record formats require special treatment, they must be distinguished by information contained in the data records.

Header: A literal "ICE" is entered in columns 8-10.

An application of a merge Input Coding Element in a variable merge program is illustrated in Figure 17.

Output Coding Element (OCE)

Attention is directed to the sort Output Coding Element. The rules and conventions for the merge Output Coding Element are identical to those of the sort Output Coding Element, except that symbol #POP is not available for fixed merges which have no output routine generated, because such programs have no need for it.

Squeeze Coding Element (SCE)

Attention is directed to the sort Squeeze Coding Element. The rules and conventions for the merge Squeeze Coding Element are identical to those of the sort Squeeze Coding Element.

The merge SCE is not executed for records with duplicate keys from the same file. The purpose of merge SCE is to process records with duplicate keys coming from separate files. To insure processing of all records with duplicate keys, the SCE should be included in the sort program which creates the sequenced file to be merged.

Label Coding Element (LCE)

The Label Coding Element is written when input label checking is desired in a variable merge. Without LCE the program halts after typing out input labels and it is the operator's responsibility to verify them. LCE delivers expected labels to the merge so that a program label check is possible.

The merge calls to the first instruction of the LCE with an SPB instruction specifying modification word 1. The merge is re-entered with the instruction BRU 1, 1.

ENTRY/EXIT CONDITIONS

The address of the previous input file label is the A-register when LCE is called.

When control is returned to the merge, the A-register contains the address of the next input file label. This address will not be the same as the "previous label" address originally in register A.

The A-register contains zero the first time the merge calls the LCE.

SPECIAL MERGE SYMBOL

#NUM	The symbolic address of the number of files to be merged.

LCE is executed each time the merge prepares to read a file for the first time.

CONVENTIONS

The "previous label" must not be changed or destroyed by LCE.

The merge always assumes that the label is five words; the first three conventionally contain identification and the other two contain the date produced.

Header: A literal "LCE" is entered in columns 8-10.

Figure 17 summarizes the data flow of a variable merge program with User Coding Elements.

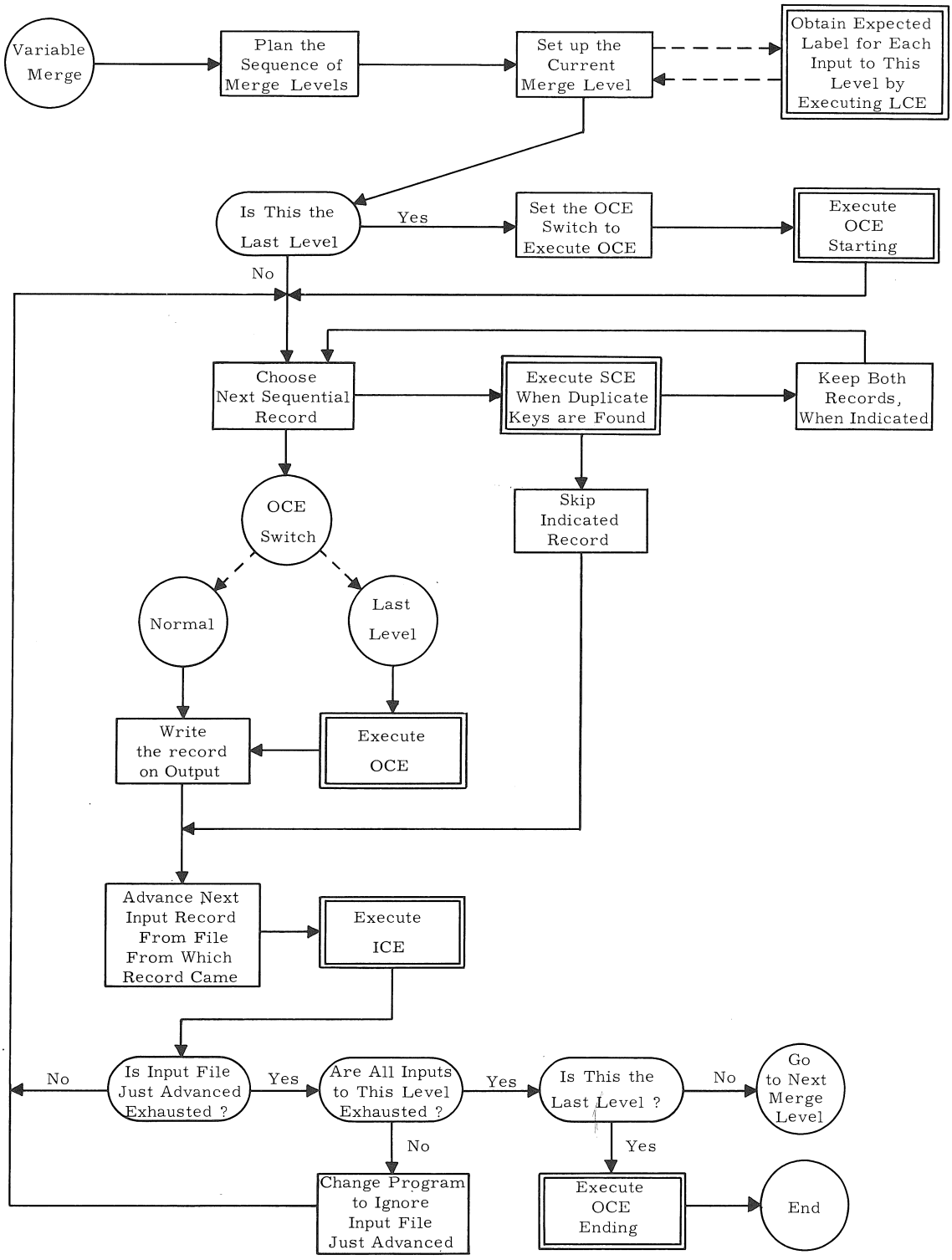


Figure 17. Flow of Variable Merge with ICE, OCE, SCE, and LCE

VI. FORWARD SORT AND MERGE TIME REQUIREMENTS

FORWARD EFFICIENCY FEATURES

Built-in features of the FORWARD Sort/Merge Generator permit the user of the GE-215/225/235 to generate optimized sort and merge object programs. Some of the design features which contribute to the efficiency of the FORWARD generated object program are presented below.

Generated FORWARD object programs are tailored to the job at hand, and contain no redundant procedures or coding. This results in maximizing the amount of memory available for work storages and buffers, and at the same time minimizing computing time.

GE-215/225/235 read/write/compute simultaneity is used to advantage throughout FORWARD object programs, and even in the generator itself, to save time.

Data moves in memory are avoided whenever possible, to keep computing time at a minimum.

Double-length instructions are used in FORWARD object programs whenever their use can improve timing.

If the sequence of records on the input tape is such that groups of records appear in the order in which they should occur in the output, presort output strings will contain more records; fewer strings will result; and fewer rotary-merge passes will be required. Thus if the input is nearly in the output order, one rotary-merge pass will often suffice, and sometimes the presort can order the output completely. Sort times are always decreased by such "favorably biased" input.

Tournament-type comparison routines are used throughout FORWARD object programs to minimize processing time.

The tournament in the presort maximizes string lengths and thereby reduces sorting times by arraying as many records in tournament storage as available memory will hold.

Collation tape blocking factors are automatically maximized to make optimum use of available memory so tape times can be minimized.

The FORWARD rotary-merge collation technique minimizes the amount of tape processing required in a sort; efficient sorting can be accomplished with as few as three collation tapes.

Although tapes used by sorts and variable merges are assigned on program generation cards, they may be reassigned when the object program is executed.

SYSTEM DESIGN CONSIDERATIONS

Although fully optimized sort and merge object programs may be produced by FORWARD for a wide variety of system configurations and data file designs, system design has a significant influence on program efficiency. Planning for the most efficient sort and merge programs for any application should reflect the factors listed below, which affect the time requirements. Within inherent limitations, consideration of each of the factors for system design decreases the amount of time required for the sort and merge object programs.

The higher the tape speed and density, the faster the sort and merge object programs.

The more collation tapes available for the sort, the fewer rotary-merge phases.

The more memory available to the presort, the longer will be the expected string length. Thus fewer strings are expected, and fewer rotary-merge phases will be needed.

Two controllers permit faster sorting than one controller.

The smaller the key, the faster the presort, since fewer comparisons must be made to find the "smaller" of two records.

Even numbered key sizes and even numbered record sizes (in words) lead to faster sorts and merges by permitting double length operations in moving and comparing routines.

The smaller the record (in number of words), the less tape movement required for a given number of records. Whenever records with duplicate keys can be combined or eliminated and when many records with duplicate keys are expected, a Squeeze Coding Element will speed up the sort. The more phases expected, the less an SCE will help.

The larger the blocks are on collation tapes, the less time will be spent per record for starting and stopping tapes. Time gains from using larger blocking factors in a merge or on sort collation tapes will become small when blocking factors are ten or more. (Block sizes of two to three hundred words are convenient for most data processing programs.)

If a program is computer limited, no time gain is made by increasing tape time. This occurs when the blocking factors of the input tapes are decreased. A time gain might be made by improving the computing time. For example, the only time reblocking speeds up a program is when the program is already tape-limited and block sizes can be increased. The concept of tape limitation arises from the fact that when given the record sizes, blocking factors, etc., for each tape file, the program cannot run in less than tape time. The important fact is that tape limitation does not necessarily imply minimum time.

ESTIMATING FORWARD SORT AND MERGE TIME REQUIREMENTS

Sort time estimates may be calculated in either of three different ways. The quickest and easiest way is to use the timing tables presented in the following paragraph on Rapid Estimation of FORWARD sort times. A second way, employing the computer itself, is to use the GE-215/225/235 library routine CD225A9.001, Sort Time Calculation Program. A third way is to employ the algebraic timing formulas. Merge time requirements may be estimated by following the procedures for timing a merge of single sorted reels and calculating the total time, considering the number of reels, tape rewinding, and reel changing time.

Rapid Estimation of FORWARD Sort Times

FORWARD sort times may be estimated easily and rapidly by use of the timing factors presented in the tables and graphs in Appendix B. The accuracy of the time factors is plus or minus 20 percent. Time factors are provided in the table for representative BCD record sizes using optimum tape blocking. However, sort time estimates for any record size ranging between 5 and 100 words may be obtained by interpolation between the nearest values shown in the tables. Time estimates derived in this manner will be correct within the estimated accuracy of plus or minus 20 percent.

The method of rapid estimation of sort time for BCD records may be used for 20-bit binary-mode records if the actual record size (in words) is multiplied by 4/3.

If the data to be sorted exceeds one full magnetic tape reel, the time should be calculated for each reel and then totaled. For a total estimate of sort time, this total time should be added to the time required to merge the output along with any time necessary for rewinding and changing of reels. If it is intended to rewind the input tape and replace it with a blank tape during the sort, time should be allowed to change the input tape. To calculate the rewind time, assume that a full reel rewinds in four minutes.

The sort time factors given in the tables are based on records with key sizes of four words or less. However, if the key exceeds four words, the sort time may be calculated by adding $\frac{k-4}{20}$, where k is the key size in words, to the factor obtained from the table.

It should be noted that the successive values in a particular row or column of the tables do not always change at a smooth rate. The reason for this is that many aspects of the sort problem can cause the sort time to increase or decrease sharply. As an example, one added record can sometimes add an extra collation pass; similarly, optimum blocking is harder to achieve with large records. Thus the time estimate is actually an uneven function of many variables.

FORWARD Sort and Merge Timing Formula

CONSTANTS USED:

b_1 = number of records in each input block.

b_2 = sort blocking factor

c = nominal character transmission rate per second.

d = average time to start and stop in seconds.

g = average length of record gaps in inches.

k = number of words in sort key.

m = memory size in number of words (4096 or 8192).

n = total number of records being sorted per cycle.

r = rewind speed in inches per second.

u = number of characters per tape word (4 if binary or 3 if BCD).

v = nominal tape speed in inches per second.

w = number of words in record.

PRELIMINARY CONSIDERATIONS

If the sort is expected to produce only one reel of output, the calculated value of T, the time to sort one output reel is the final sort time estimate. If the output is expected to exceed one full reel, the sorting should be performed using one reel at a time, and a merge should follow. Calculate the time to sort each reel and add the time for the merge. (See the paragraph which follows on timing merge of single sorted reels.) The maximum number of records on one full reel (assuming 2300 feet of tape) may be determined by the following formula:

$$n = (27600b_2) \div (\frac{uv}{c} \cdot b_2 \cdot w + .75),$$

T_2 is the time to rewind input and replace with blank tape. If it is not desired to use all the tape handlers on the computer for sorting, it is unnecessary to calculate T_2. Otherwise, it must be calculated and included in the total time.

CALCULATIONS

Schedules which are referred to are on succeeding pages.

Presort time (T_1)

Without MOV Command $a = .00075k + .00021w + .004$
With MOV Command $a = .00075k + .00007w + .004$

$$t = \frac{u}{c} \cdot w + \frac{d}{b_1}$$

Find the value of y, the nonsimultaneity factor, in Schedule 1.

If $y \cdot t \geq a$, $T_1 = n \cdot (y\ t + \frac{.01}{b_1})$

Otherwise, $T_1 = n \cdot (a + \frac{.01}{b_1})$

Time to rewind input and replace with blank tape (optional) (T_2)

$$T_2 = n \cdot \frac{1}{r} \cdot (\frac{uv}{c} \cdot w + \frac{g}{b_1}) + 30$$

Add 30 seconds if it is desired to rewind the program tape and replace with a blank.

Collation Time (T_3)

Find the value of B, the maximum sort block size, from Schedule 2. The actual block size will be the largest multiple of w which is less than or equal to B, say $b_2 \cdot w$. Then b_2 is the sort blocking factor.

Tournament size (Z):

$$Z = \frac{m - 1550 - 2(b_1 + b_2) - w}{w + 20}$$

Expected number of strings: $s=\frac{n}{2Z}$; round to the next higher integer and use s to find p, the expected number of phases, from Schedule 4. Find f, the average phase factor, in Schedule 3.

$$T_3 = n \cdot p \cdot f\left\{ y \left(\frac{u}{c} \cdot w + \frac{d}{b_2}\right) + \frac{1}{r}\left(\frac{uv}{c} \cdot w + \frac{g}{b_2}\right) \right\}$$

Without MOV Command $a = .00045k + .00018w + .003$
With MOV Command $a = .00045k + .00004w + .003$

"a" should be used in the place of $y(\frac{u}{c} \cdot w + \frac{d}{b_2})$ if "a" is greater than:

$$y\left(\frac{u}{c} \cdot w + \frac{d}{b_2}\right).$$

Total time in seconds (T)

$$T = T_1 + T_2 + T_3$$

If User Coding Elements are included, add the time to perform them.

POSSIBLE SIMPLIFICATIONS

The timing formula may be simplified considerably by precalculating certain quantities and assuming certain values (which are expressed as variables in the formula) to be constant. Thus the following might be assumed:

$v = 75$

$g = .75$

$d = .015$

$1/r = .0067$ when $r = 150$

$\frac{3}{c} = \begin{cases} .000200 & (15{,}000 \text{ character/second tapes}) \\ .000072 & (41{,}700 \text{ character/second tapes}) \end{cases}$

$\frac{3v}{c} = \begin{cases} .0150 & (15{,}000 \text{ character/second tapes}) \\ .0054 & (41{,}700 \text{ character/second tapes}) \end{cases}$

$$\frac{4}{c} = \begin{cases} .000266 & \text{(15,000 character/second tapes)} \\ .000096 & \text{(41,700 character/second tapes)} \end{cases}$$

$$\frac{4v}{c} = \begin{cases} .0200 & \text{(15,000 or 22,500 character/second tapes)} \\ .0072 & \text{(41,700 character/second tapes)} \end{cases}$$

If b_1 is not known, use b_2 as an estimate, as calculated for collation time. When $b_1 = b_2$, the following simplifications occur:

For Presort Time (T_1) and Collation Time (T_3):

$$t = \left(\frac{u}{c} \cdot w + \frac{d}{b_1}\right) = \left(\frac{u}{c} \cdot w + \frac{d}{b_2}\right)$$

For the time required to rewind input and replace with blank (T_2) and Collation Time (T_3).

$$\frac{1}{r} \cdot \left(\frac{uv}{c} \cdot w + \frac{g}{b_1}\right) = \frac{1}{r} \cdot \left(\frac{uv}{c} \cdot w + \frac{g}{b_2}\right)$$

If it is assumed that $b_1 = b_2$, the following alternate method may be used to calculate T:

$$a = .00025k + .0004w + .003$$

$$t = \frac{u}{c} \cdot w + \frac{d}{b_1}$$

$$R = \frac{1}{r} \cdot \left(\frac{uv}{c} \cdot w + \frac{g}{b_1}\right)$$

$$T = n \cdot \left\{a + \frac{.01}{b_1} + p \cdot f\,(y\,t + R)\right\}$$

with p and f determined as described for Collation Time (T_3). If it is desired to add T_2,

$$T_2 = n \cdot R + 30.$$

TIMING A MERGE OF SINGLE SORTED REELS

Find the value of y, the nonsimultaneity factor, in Schedule 1.

Find the value of Q, the merge factor, in Schedule 5. For fixed merges, Q = 1.

Estimated time in seconds is

$$T = Q \cdot n \cdot \left\{ y \left(\frac{u}{c} \cdot w + \frac{d}{b_1}\right) + \frac{1}{r} \left(\frac{uv}{c} \cdot w + \frac{g}{b_1}\right) \right\}$$

SCHEDULE 1

(nonsimultaneity factor, y)

Nominal Tape Speed	Y, One Controller	Y, Two Controllers
15 kc	2.0	1.5
41.7kc	2.2	1.7

SCHEDULE 2

(maximun sort block sizes, B)

No. of TAPES	3	4	5	6	7	8
Memory 4,096	500	375	300	250	210	180
8,192 or 16,384	1150	875	700	575	500	435

The actual sort block size is the largest multiple of w which is less than or equal to the maximum block size.

SCHEDULE 3

(average phase factors, f)

Number of Tapes In Sort	f
3	.75
4	.67
5	.62
6	.60
7	.59
8	.58

SCHEDULE 4

(expected number of phases, p)

Number of Phases	Number of Tapes					
	3	4	5	6	7	8
1	2	3	4	5	6	7
2	3	5	7	9	11	13
3	5	9	13	17	21	25
4	8	17	25	33	41	49
5	13	31	49	65	81	97
6	21	57	94	129	161	193
7	34	105	181	253	321	385
8	55	193	349	497	636	769
9	89	355	673	977	1261	1531
10	144	653	1297	1921	2501	3049
11	233	1201	2500	3777	4961	6073
12	377	2209	4819	7425		
13	610	4063				
14	987					
15	1597					
16	2584					
17	4181					

Find the column headed by the desired number of tapes. Find the smallest value in the chosen column which exceeds s; the corresponding number in the leftmost column is p, the expected number of phases.

SCHEDULE 5

(merge factors)

Number of Input Reels	Number of Merge Tape Handlers									
	3	4	5	6	7	8	9	10	11	12
2	2	2	2	2	2	2	2	2	2	2
3	5	3	3	3	3	3	3	3	3	3
4	8	6	4	4	4	4	4	4	4	4
5	12	8	7	5	5	5	5	5	5	5
6	16	11	9	8	6	6	6	6	6	6
7	20	13	11	10	9	7	7	7	7	7
8	24	16	14	12	11	10	8	8	8	8
9	29	18	16	14	13	12	11	9	9	9
10	34	22	18	17	15	14	13	12	10	10
11	39	25	21	19	17	16	15	14	13	11
12	44	29	23	21	20	18	17	16	15	14
13	49	32	25	23	22	20	19	18	17	16
14	54	36	28	26	24	23	21	20	19	18
15	59	39	30	28	26	25	23	22	21	20
16	64	44	32	30	28	27	26	24	23	22
17	70	46	36	32	31	29	28	26	25	24
18	76	50	39	35	33	31	30	29	27	26
19	82	53	42	37	35	33	32	31	29	28
20	88	57	46	39	37	36	34	33	32	30

The desired factor is found where the horizontal row corresponding to the expected number of input reels intersects the vertical column corresponding to the total number of merge tape-handlers (including merge output tape-handler).

VII. OPERATING INSTRUCTIONS FOR FORWARD SORT/MERGE GENERATOR

FORWARD Sort/Merge Generator program (CD225G1.002) is normally provided as a binary object card deck. Prior to program use, this card deck must be loaded on magnetic tape. (See Figure 18.) When loading is completed, the magnetic tape becomes the FORWARD Sort/Merge Generator systems tape. The procedure for loading the card deck on magnetic tape is as follows:

Remove the "Remarks" cards from the front of the deck (identified by end-printed numbers 1-8). Remove the FORWARD Sort/Merge Generator Call card (identified by end-printed number 1 0010).

Mount a reel of scratch tape on tape handler 0, priority control channel 1. This is used as a scratch tape. Mount another reel of scratch tape on tape handler 1, priority control channel 1. This tape becomes the FORWARD Sort/Merge Generator systems tape.

If a printed listing is desired while loading the cards on tape, leave console switch 18 in the normal position. If printing is to be suppressed, set switch 18 to the down position.

Load the card deck and start the computer.

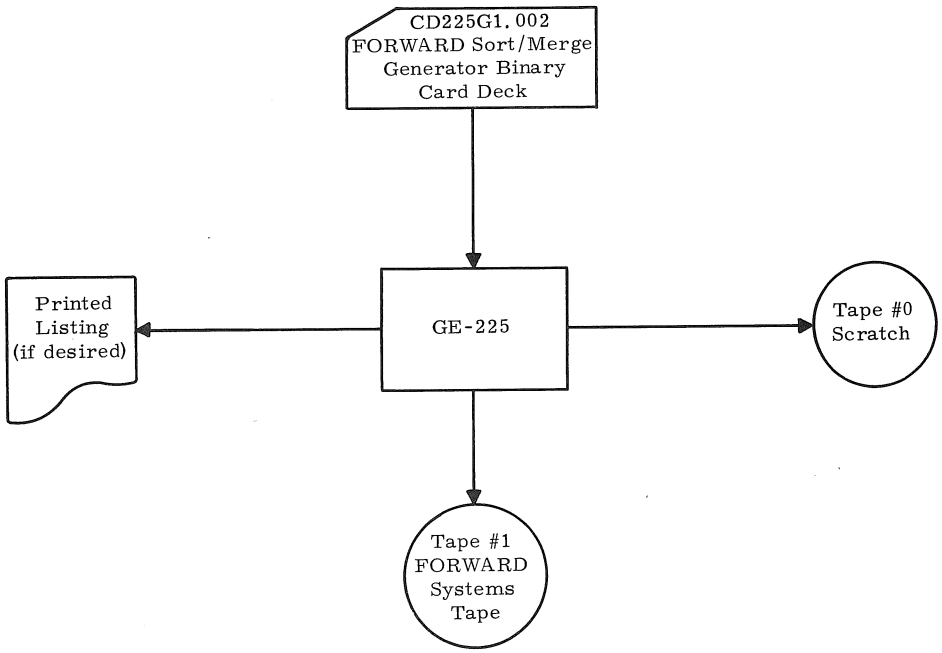


Figure 18. Loading FORWARD Generator Binary Card Deck on FORWARD System Tape

Console typewriter messages:

BAD RD XXXXX, where XXXXX is the card number, signifies a bad card read. The cards continue to be processed to tape even though a bad read is indicated.

P1 T1 is typed out upon successful completion of loading the cards on magnetic tape.

System Hardware Requirements

4096 word memory (or larger)

Printer on priority control channel 6

Card reader

Console typewriter turned on

Blank magnetic tapes on tape handlers 3 and 4 connected through priority control channel 1.

FORWARD Generator System tape mounted on any tape handler (as specified in the FORWARD Call card). See Figure 19 for the System Configuration and Data Flow for FORWARD Sort or Merge Program Generation and Assembly.

Generator Input

(See Figures 5 & 6 for the card arrangement for the FORWARD Sort/Merge Generator Input.)

FORWARD Call Card (See FORWARD Call Card Description which follows.)

Parameter cards and any desired User Coding Element cards (See Section IV.)

Two blank cards.

Generator Output

Symbolic sort or merge program on priority control channel 1, tape handler 3.

Parameter cards and logical error notes listed on the printer. (See Figures 20, 21 and 22.)

Generator identification, operating instructions, and condensed error notes on the console typewriter. (See Figure 23.)

FORWARD Call Card Description

Columns 77 and 78 of the Call card are punched by the user to assign the FORWARD system tape channel and tape number. Punch the desired channel number in column 77 and the tape handler number in column 78 in Hollerith code. For example, a punch in row 2 of column 77 and row 6 of column 78 would indicate to the Call card that the FORWARD system tape is mounted on channel 2, tape handler 6. These columns must be punched with a valid channel and tape assignment; blanks or punches on rows 8 or 9 will be assigned a zero value.

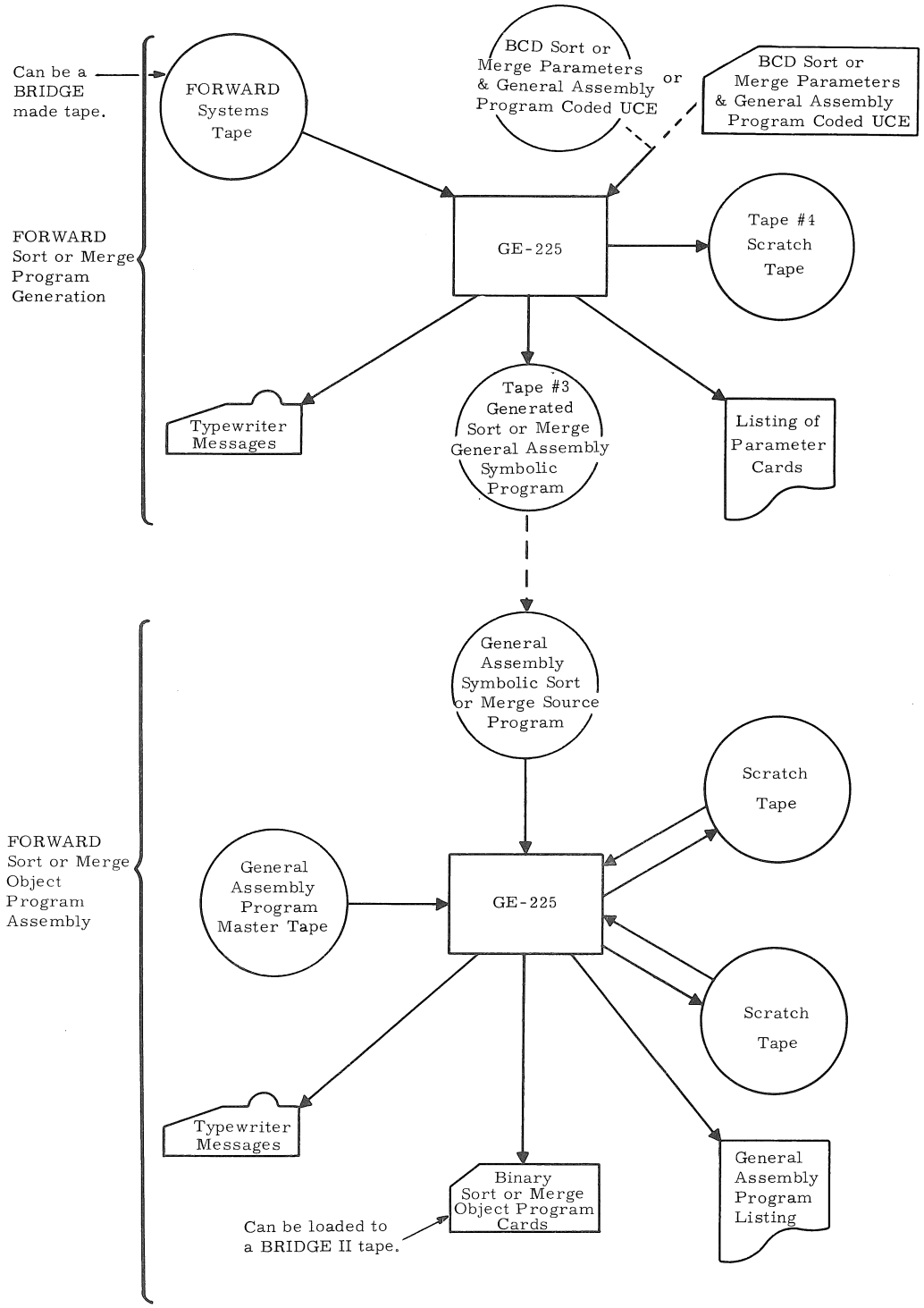


Figure 19. FORWARD Sort or Merge Program Generation and Assembly

```
PROGRAM CARD      SRT 8  025   10 YNYD          SORT TEST
INPUT CARD        INPUT  Y Y 020 11             SORT INPUT      100
I-FILE CARD       I-FILE                        INPUT1,INPUT2,INPUT3
ICE CARD                 ICE
OUTPUT CARD       OUTPUT Y Y 025 13             SORT OUTPUT     200
O-FILE CARD       O-FILE                        OUT1,OUT2,OUT3
OCE CARD                 OCE
SORT CARD         SORT Y  T  N    125Y          13,14,23,24,25
C-FILE CARD       C-FILE                        COLLATION TAPES
SCE CARD                 SCE
KEY CARD          KEY                           08,05,09,03,04,00,07,01,02,06
END OF AUDIT
```

Figure 20. Printout During Sort Program Generation

```
PROGRAM CARD      MRG 8  040   04 Y ND          MERGE 1
INPUT CARD        INPUT  Y Y 010
ICE CARD                 ICE
OUTPUT CARD       OUTPUT Y Y 020 17             END OF LINE     150
OCE CARD                 OCE
SCE CARD                 SCE
MERGE CARD        MERGE   04      100Y
M-FILE CARD       M-FILE     12                 AAAAAAAAA TODAY
M-FILE CARD       M-FILE     13                 BBBBBBBBB TODAY
M-FILE CARD       M-FILE     14                 CCCCCCCCC TODAY
M-FILE CARD       M-FILE     15                 DDDDDDDDD TODAY
END OF AUDIT
```

Figure 21. Printout During Fixed-Merge Program Generation

```
PROGRAM CARD      MRG 8  040   04    D          GRUMBLE
INPUT CARD        INPUT  Y N 010                ANY INPUT TODAY
OUTPUT CARD       OUTPUT Y N 010 15             U-NAME-IT BUDDY
MERGE CARD        MERGE                         12,13,14
END OF AUDIT
```

Figure 22. Printout During Variable Merge Program Generation

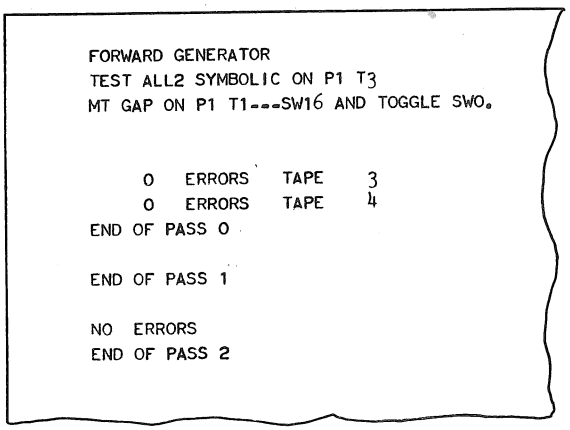


Figure 23. Generator Typeout Indicating Successful FORWARD Sort or Merge Program Generation and General Assembly Program Assembly

Console Typewriter Messages and Associated Operator Action:

Message	Condition	Action
FORWARD GENERATOR	Typed when generator is loaded.	None
BAD PARAM-- RESUBMIT	Erroneous parameter values prevent generating desired sort or merge program.	Correct the parameter cards and resubmit.
BAD PARAM-- SW19 TO IGNORE	The generator has detected parameter errors but is willing to guess at correct values.	Either set SW10 and toggle SW0 to continue, or correct the parameter cards and resubmit.
NO END CD	End card is missing in last User Coding Element named on printer.	Insert END card and resubmit.
Program ID SYMBOLIC ON P1 T3---MT GAP ON P1 T1--SW16 AND TOGGLE SW0	Generation is complete.	Indicated action will cause FORWARD to call General Assembly Program to assemble generated program.
Other messages	Hardware errors on parameter errors.	See Section VIII, FORWARD Sort/Merge Generator Error Detection.
Error All	Other than blank, "N" or "Y" punch in column 16.	The generator will assume that no API feature is to be included in the generated object program.

Use of API Object Program Option

The computer will be set up according to the parameters that were input to the FORWARD Generator. The loader must be the General Absolute Loader (CD225B1.013R) with the API module. Cards will be loaded in the following order:

1. The General Absolute Loader
2. Priority Programs
3. The Sort or Merge Object Program

If the priority programs are assembled at location #CS or #SCE during generation time, their priority indicator cards (ORG 0) must be pulled and placed before the sort object program at load time (see the writeup for the routine CD225B1.013R).

VIII. FORWARD SORT/MERGE GENERATOR ERROR DETECTION

The FORWARD Sort/Merge Generator detects three classes of errors: invalid or missing information in the input which can be estimated by the generator; invalid or missing information which cannot be estimated (with any assurance); and hardware failure.

Hardware failure can produce errors which may be detected and corrected as follows:

Card Reader - When a bad read is detected, the generator types "BAD READ" and goes into a programmed loop. When this occurs program generation must be restarted.

Printer - The generator will loop executing a "branch on printer ready" or "branch on printer error condition." To continue, correct the printer's complaint and depress the "manual clear" button on the printer controller.

Magnetic Tapes - Conventional error detection and correction routines are performed. If an uncorrectable error is encountered, a conventional message of the form "PX TY EZ" is typed out, and the computer halts. The generator must be restarted.

Logical errors in the input parameters produce messages on the printer. Each parameter card is printed when it is read and errors in the parameters or inconsistent parameters are listed below a printout of the name and contents of the card. All numeric fields are checked for valid range and for valid decimal digits. Channel and tape numbers are checked for duplication. Alphanumeric fields are checked for complete blanks, and a check is always made to assure that fields which should be blank (in view of prior parameters already processed) are indeed blank and vice versa. When a parameter takes a value which reflects logically on other parameters, a consistency check is made to assure that there is no conflict. The entire input is always passed through a complete error analysis, whether or not invalid or inconsistent parameters are discovered. (The analysis and interpretation of the input takes place in the first generator pass.) When any of the above checks detect an error, the field in which the error occurs is identified on the printout with a message symbol as indexed in the ID column of the Parameter Card Description Table in Section IV.

The design of the FORWARD error analysis function precludes the production of programs which will not run because of bad parametric descriptions. Most of the parameters used by FORWARD will not lead to unusable object programs, even if they are specified wrong, because the generator normally will make a guess at what is necessary for an erroneous parameter to be consistent with the remaining parameters. The actual guessing that the generator will undertake is all specified in the FORWARD Parameter Card Description Table in Section IV.

At the end of the first generator pass one of three conditions is true: Either no errors have been detected, "guessable" errors have been detected and lister, or "unguessable" errors have been detected and listed (along with "guessable" errors, if present). If no errors have been detected the generator proceeds immediately to the second pass, in which the actual object program is generated and written onto tape for input to the General Assembly Program.

If "guessable" errors have been detected, the generator causes the console typewriter to type out "BAD PARAM --- SW19 TO IGNORE." The operator may set switch SW19 to the down position and toggle the sign switch to enter the second generator pass. The generator will, in this case produce a coherent object program which will run. Only the programmer, however, can determine whether the generator has produced a satisfactory object program or whether it has produced something coherent but useless. This information may be obtained by checking the error note on the listing for the parameter in question against the Parameter Card Description Table in Section IV to see what assumption has been made.

If "unguessable" errors have been detected, the generator causes the console typewriter to type "BAD PARAM --- RESUBMIT." In this case, the input must be corrected and resubmitted. For quick reference, parameters which can contain "unguessable" errors have their ID symbols underlined in the Parameter Card Description Table in Section IV.

The printer indicates a bad parameter with a message of the form "ERROR XX" under the printout of the card in which the parameter is punched. Here XX is the two-character "ID" listed in the right-hand column of the Parameter Card Description Table in Section IV. The parameter ID's are generally in order throughout the table so that an error message may be quickly interpreted by scanning the ID column for the given value, then reading in the adjacent column to determine what constitutes an error. Progressing from right to left, the programmer can determine what guess the generator has made (or that it has refused, when the "action" column indicates "can't guess") and what correction to make if the guess is not satisfactory. When an error message can result from conflicting parameter values, the programmer should check all parameters involved.

When a rush of error messages follows a parameter card, or a group of several parameter cards, it will usually be discovered without reference to the Parameter Description Table that either a card is missing or that some cards are not in the required order.

The message "NO END CD" printed out under the Header Card title of a User Coding Element indicates that the entire input deck has been consumed in a futile search for the END card. In this case the programmer should supply the END card and resubmit the input. When a parameter card tells the generator not to expect a specific User Coding Element (other than SCE) but the User Coding Element actually does appear in the input, the generator ignores the element but lists what it ignores. Otherwise the User Coding Element will not be listed, since it appears in the General Assembly Program listing of the object program. No examination is made of the cards within a User Coding Element, except the searching for the END card.

The sample error printout shown in Figure 24 indicates that the programmer has made several mistakes in preparing his parameter cards. Although the generator would be willing to guess the correct values for the bad parameters, in a case of this type it is recommended that the cards be corrected and resubmitted. The input to the generator which produced the printout shown in Figure 20 is identical to the input which produced the printout shown in Figure 24, except that, in the latter mistakes occurred in the parameter cards. Refer to the Parameter Card Description Table in Section IV, where error codes may be found in the ID column. Every symbol in the ID column may occur in a printout after the word ERROR. The type of error may be determined by referring to the table. The ERROR ID typeout refers to an error in any of the information in the section to the left of the ID symbol. The errors indicated in the printout of Figure 24 are interpreted on the following page.

IX. OPERATING INSTRUCTIONS FOR FORWARD SORT OBJECT PROGRAM

A loader must be placed at the front of the deck, preceded by a memory resetter. The non-reset loader Binary Loader With Memory Print Option, (CD225B6.004) may be used, or any other card loader requiring no memory outside of location $(0\text{-}169)_{10}$. (The binary cards preceding the first transfer card represent the presort subprogram, and include a symbolic input/output for magnetic tape subroutine, along with the other subroutines and constants used throughout the sort. The remaining binary cards represent the rotary-merge subprogram.) (See Figure 25.)

The rotary-merge subprogram is automatically written on the second collation tape for later use. An option is provided which permits changing the collation tape list when the sort is executed. To use this option the operator must insert a tape list card after the first transfer card in the binary deck but before the rotary-merge subprogram. The format of the decimal tape list card is as follows:

Columns 1 - 2	First priority control channel and tape addresses
Column 3	Blank
Columns 4 - 5	Second priority control channel and tape addresses
Column 6	Blank
Columns 7 - 8	Third priority control channel and tape addresses
Column 9	Blank
Etc.	

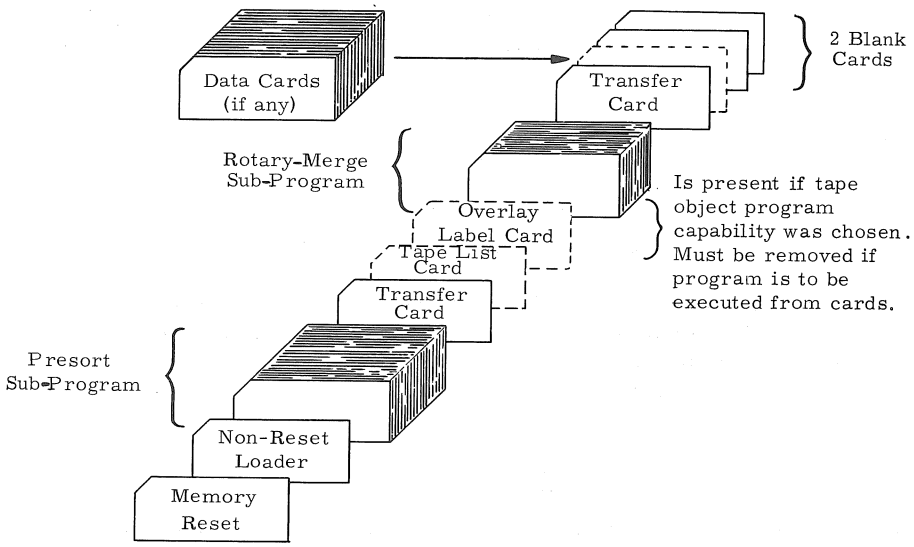


Figure 25. FORWARD Sort Object Program Input Card Deck Arrangement

Up to as many tapes as were specified on the sort parameter card may be indicated on the tape list card. If necessary, as few as three tapes may be included. If data cards are used during the presort subprogram run, they must be placed after the transfer card which follows the rotary-merge subprogram cards. If the output parameter card indicated a specified output tape, the first priority control channel and tape mentioned in the tape list card receives the output. Normal use of the tape list card is to reduce the number of collation tapes used by a generated sort. Tapes and channels specified need have no relation to the original list on the sort card, so that the tape list card can be used simply to change the collation tape assignments.

The tape list card also performs tape date updating. The input date update option is used to avoid label check failure on sort input tapes when the tape date has been changed since program generation. The output date update option permits the date written on the output tape to be changed at operation time to a date different from the one specified at generation time.

The update options are not applicable to a fixed merge since it does not use a tape list card. The input data columns are not used for a variable merge since it uses no label check on inputs. The tape list card with its date update field, includes three options. The following is a summary for the entire card, specifying the interdependency requirements.

Column	Description	Format	Requirement
		TAPE LIST OPTION	
1-24	1. Sort Collation List 2. Merge Inputs	DD△DD△DD... DD△DD△DD...	If the tape object program option was chosen, 79-80 must be entered also.
		DATING OPTION	
31-42	SORT DATE	Input Output DDDDDDDDDDDD	Columns 1...24 must also be entered even if no change is desired.
37-42	MERGE DATE	Output DDDDDD	
		OBJECT PROGRAM OPTION	
79-80	Instruction tape priority control channel and handler or change if card option. (See PS "A" on Tape Object Program.)	DD or "CD"	No other fields on the tape list card need be used.

It should be noted that the tape list data in columns 1...24 must be entered correctly if the date option is used. The data in columns 31-42 for sort and 37-42 for merge will all be used if any nonblank character is in columns 34-36 for the sort and columns 37-39 for the merge.

Collation tape handlers should have blank 2400 foot reels of magnetic tape in good condition mounted before the sort is started, and collation tapes should be positioned at the load point, as should the input tape. (A check rewind on the collation tapes is performed immediately after the program is loaded, following interpretation of the tape list card, if one is provided.)

A rerun checkpoint is established at the end of every pass of the sort, and whenever an intermediate output tape or file is closed. The checkpoint is identified by a conventional typewritten message. To rerun a pass (or rotary-merge phase), use Rerun 2 Program (CD225J5.000) and follow the procedures given in Symbolic Tape I/O System (CD225E2.001). Only the last rerun checkpoint indicated on the typewriter log may be used, because in general tape contents have been changed since the passing of previous checkpoints.

Figure 26 illustrates a sample typeout of typical operating messages which will be typed during execution of a sort object program. The following list shows console typewriter messages and operator actions required during execution of the sort object program. The first set of messages in the list occurs when applicable, immediately after the program is loaded. The second set of messages occurs after processing has begun.

Initial Message	Action
Program Identification	None
TAPES OK	Toggle SW0 if a stored collation tape list is to be used, or set SW19 and toggle SW0 if a tape list card is to be used.
YES (after TAPES OK)	None - confirms that SW0 was toggled without SW19 being set.
NO (after TAPES OK)	None - confirms that SW0 was toggled with SW19 set.
CARD OPTION	None - confirms that the object program is being run from cards, in response to "CD" punched in col. 79-80 of the tape list card.
TAPE OPTION INST TAPE PXTY	None - confirms that the object program is being run from channel X tape Y, as specified in col. 79-80 of the tape list card.
BAD RD	(Program is trying to read either the tape list card or the rotary-merge subprogram.) Backspace card reader and toggle SW0.
ERROR	(Program is trying to interpret the tape list card.) Tape list card contains illegal format, non-existent channel or tape numbers, or there are fewer than three tapes. Correct and resubmit.
OUTPUT REEL	Set the switch corresponding to the desired reel number for the output label and toggle SW0. For example, set SW6 to specify reel 006. The first switch set (from the left-hand side of the console) specifies the output reel; if SW0 only is toggled, the output reel number is set to 001.

Three digit number (after OUTPUT REEL)	Confirms the output reel number which was entered via the console switches.
REBLOK	Toggle SW0 to continue. (Memory requirements for the desired number of collation tapes exceed available memory; the last tape in the list is dropped if the program is continued.)
NO CAN DO	Memory is exceeded even if only three collation tapes are used. Modify the source program to reduce the memory requirements (normally this is done by cutting down OCE) and resubmit to the generator.

Processing Messages	Action
Conventional magnetic tape messages	Follow the procedures described in Symbolic I/O System (CD225E2.001).
PX TY END	"X" and "Y" indicate the input channel and tape; set SW19 and toggle SW0 to accept another reel of input; or toggle SW0 only to end input. This option is presented at each end of reel or end of file, giving the operator control of overall input procedures.
END-PRSRT	Presort is finished. Toggle SW0 to read in rotary-merge. If the program is discontinued at this point, Rerun 2 Program (CD225J5.000) must be used to resume processing.
END SORT	The sort object program run is completed. The last output tape identified on the typewriter log is the final sorted output, unless OCE has made other provisions for output. Toggle SW0 to load the next card program (reads one binary card into 0 and branches to 0).
MULTI-CYCLE NNN	The output tape of the current cycle begins rewinding. NNN is the number of the cycle just completed. Remove the tape when it has rewound and replace it with a blank tape. Toggle console switch 0 to start the next cycle of the sort.

```
TEST ALL
TAPES OK YES
OUTPUT REEL 001

                    P1 T1 001 RANDOM NUMBERS
                    P1 T1 END
                                        P1 T2 001TEST OUT1      RP
                                        P1 T3 001TEST OUT1      RP
                                        P1 T4 001TEST OUT1      RP

END-PRSRT

                                        P1 T5 001TEST OUT1      RP
                                        P1 T4 001TEST OUT1      RP
                                        P1 T2 001TEST OUT1

END SORT
```

Figure 26. Console Typewriter Typeout During Execution of Sort Object Program

X. OPERATING INSTRUCTIONS FOR FORWARD MERGE OBJECT PROGRAM

A loader must be placed at the front of the binary card deck. Binary Loader with Memory Print Option (CD225B1.004) or any other card loader requiring no memory outside of locations $(0\text{-}169)_{10}$ may be used. The binary deck is all loaded at once. (See Figure 27.)

For variable merges (not fixed merges) an option is provided to change the input tape list when the merge is executed. To use this option, the operator must add a tape list card immediately after the transfer card in the binary deck. The format of the decimal tape list card is as follows:

Columns 1 - 2	First priority control channel and tape addresses
Column 3	Blank
Columns 4 - 5	Second priority control channel and tape addresses
Column 6	Blank
Columns 7 - 8	Third priority control channel and tape addresses
Column 9	Blank
Etc.	

Up to as many magnetic tapes as were specified on the merge parameter card may be indicated on the tape list card; at least one tape must be specified when a tape list card is used. The

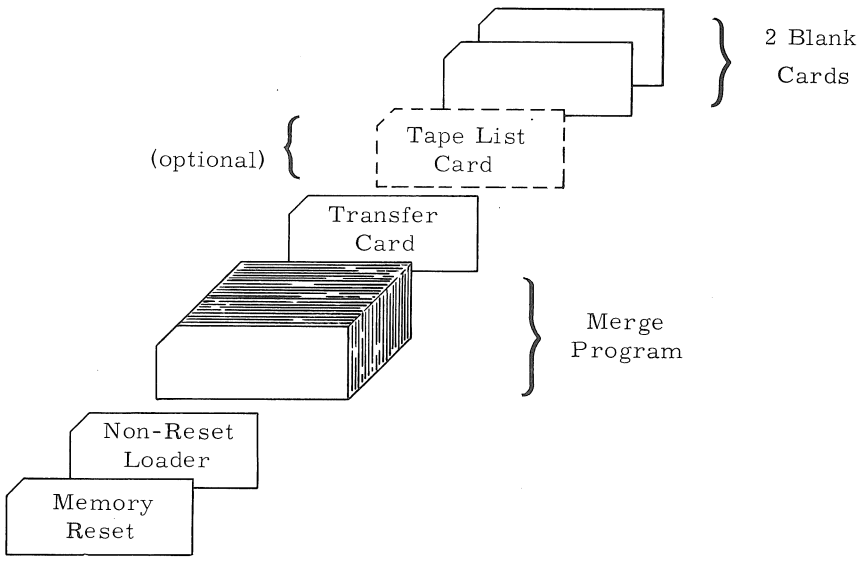


Figure 27. FORWARD Merge Object Program Input Card Deck Arrangement

normal use of the tape list card is to reduce the number of tape handlers used by the merge; tapes and channels specified need have no relationship to the original list on the merge card, so the tape list card can be used simply to change the collation tape assignments. The tape list card may be used also to update tapes. (See Chapter IX.)

All tapes should be positioned at load point for merge processing.

A rerun checkpoint is established at the end of each intermediate output reel. The checkpoint is identified by a conventional typewritten message. To rerun, use Rerun 2 program (CD225J5.000) and follow the procedures given in Symbolic I/O System (CD225E2.001). In a variable merge the program can be discontinued and later restarted at the end of each pass. As an example, if six files are being merged with only four input tape handlers, the first pass is a three-way merge, and the second is a four-way merge of the three remaining input files with the first pass output. The operator could discontinue the program when the first pass (three-way merge) is completed. At any later time, the operator can resume processing by simply initiating a four-way merge. A merge which does not use intermediate output does not have rerun capability because it is essentially a single pass.

Variable merges automatically plan the optimum procedure for merging any number of input files. The operator specifies the number of files to be merged via the console switches according to the instructions typed out on the console typewriter. If the number of input files exceeds the available number of input tape handlers, more than one pass is required. In this case, the next to the last pass may use fewer than the maximum number of input tape handlers in order to achieve overall optimization. At the beginning of each pass, the variable merge types instructions for mounting tapes on the appropriate input tape handlers. If LCE is included, input labels are typed out with the tape mounting instructions. If LCE is not included, only the channel and tape handler numbers are typed out. (See Figure 28.) When tape mounting instructions have all been typed out on the console typewriter, the computer halts to permit the operator to do the required tape mounting. Since programmed label check is omitted on the first tape of each file when LCE is not provided, it is the operator's responsibility to verify input labels which are typed out when each file is opened. With or without LCE, conventional label checking is performed on the second and succeeding tape reels of each input file.

The operator can help to optimize a variable merge whenever two or more merge passes are required. It is always best to submit first the files that have least amount of data. (For large files, submit first the files with the fewest reels.) If relative file sizes are known, considerable merge time can be saved by applying this rule. As an example, in a case where a merge program is assigned four input tape handlers to merge files A through H, which have the numbers of records listed below, the merge requires three passes: a 4-way pass, a 2-way pass, and a final 4-way pass.

A	20,000 records
B	12,000 records
C	6,000 records
D	5,000 records
E	2,000 records
F	600 records
G	400 records
H	50 records

```
GRUMBLE
TAPES OK YES
FILES
006
MT P1 T2
MT P1 T3
MT P1 T4

                (A) P1 T2 001AAAAAAAAA TODAY
                (B) P1 T3 001BBBBBBBBB TODAY
                (C) P1 T4 001CCCCCCCCC TODAY
                                  (X) P1 T5 101U-NAME-IT BUDDY     RP
                                      (REMOVE)
TOGGLE SWO TO CONTINUE
MT P1 T2
MT P1 T3

                (D) P1 T2 001DDDDDDDDD TODAY
                (E) P1 T3 101U-NAME-IT BUDDY
                                  (Y) P1 T5 201U-NAME-IT BUDDY     RP
                                      (REMOVE)
TOGGLE SWO TO CONTINUE
MT P1 T2
MT P1 T3
MT P1 T4

                (F) P1 T2 001EEEEEEEEE TODAY
                (X) P1 T3 001FFFFFFFFF TODAY
                (Y) P1 T4 201U-NAME-IT BUDDY
                                      P1 T5 001U-NAME-IT BUDDY

END MERGE
(FILES TO BE MERGED ARE A, B, C, D, E, F.
 FILES X AND Y ARE INTERMEDIATE OUTPUT.)
```

Figure 28. Console Typewriter Typeout During Execution of Variable Merge Object Program

The typed out instructions on the console typewriter inform the operator of the number of inputs to each pass. The best choice for each pass using the files listed above would be as follows:

first pass (4-way merge) - process files E, F, G, and H producing 3050 records on the output.

second pass (2-way merge) - process file D with the first pass output, producing 8050 records on the output.

final pass (4-way merge) - process files A, B, and C with the second pass output, producing the final output of 46,050 records.

This sequence of merge passes represents a fully optimized merge with a minimum number of records processed through each pass. (Since a fixed merge always processes a preset group of input files in a single pass, fixed merge timing is not affected by relative file sizes.) In

the above example, the variable merge, after planning the optimum number of files to process, identifies the tape handlers which should have tapes mounted for each pass. The operator would have completed the optimization by submitting the smallest files in the early passes.

Figure 29 shows a typical typeout received during execution of a fixed merge object program. The following list shows console typewriter messages and operator actions required during execution of the fixed merge and variable merge object programs. Asterisks preceding instructions in the Action Column denote procedures which would be performed for variable merges only.

```
MERGE  1
                              P1 T2 001AAAAAAAAA TODAY
                              P1 T3 001BBBBBBBBB TODAY
                              P1 T4 001CCCCCCCCC TODAY
                              P1 T5 001DDDDDDDDD TODAY

                                                 P1 T7 001END OF LINE

END MERGE
```

Figure 29. Console Typewriter Typeout During Fixed-Merge Object Program Execution

Messages	Action
Program Identification	None
TAPES OK	* Toggle SW0 if a stored input tape list is to be used, or set SW19 and toggle SW0 if a tape list card is to be used.
YES (after TAPES OK)	* None - confirms that SW0 was toggled without SW19 being set.
NO (after TAPES OK)	* None - confirms that SW0 was toggled with SW19 set.
CARD OPTION	None - confirms that the object program is being run from cards, in response to "CD" punched in col. 79-80 of the tape list card.
TAPE OPTION INST TAPE PXTY	None - confirms that the object program is being run from channel "X" tape "Y", as specified in col. 79-80 of the tape list card.
BAD RD	* (Program is trying to read the tape list card.) Back space the card reader and toggle SW0.
ERROR	* (Program is trying to interpret the tape list card.) The tape list card contains illegal format or nonexistent channel or tape numbers. Correct and resubmit.

Messages	Action
FILES	* Set the console switch corresponding to desired number of input files and toggle SW0. For example, set SW6 to specify 6 files. The first switch set, other than SW0, from the left-hand side of the console specifies the number of files. A positive setting is required.
PLEASE (after FILES)	* (Program is trying to determine desired number of input files; the operator has toggled SW0 without setting the switch for the required number of files.) Perform same action as specified for response to FILES message above.
Three digit number (after FILES or PLEASE)	* Confirms the number of input files specified via console switches.
NO CAN DO	Insufficient memory is available for even one input tape with the assigned blocking factor. Modify source program to reduce memory requirements (normally this is done by reducing OCE or blocking factors) and resubmit to generator.
LIMIT IS XXX (XXX a three digit number)	* None. (Memory requirements for the desired number of input files exceed available memory even when only one buffer per tape is assigned; the program continues with a reduced number of input tape handlers, indicated by the number typed; only one buffer is assigned to each file in this case.)
MT PX TY (X and Y octal digits; a list of such messages is followed by a console switch loop.)	* Mount a merge input tape on channel X, tape Y. In a merge involving more than one pass, the file to be mounted should be the one having the least data of those not yet merged. Toggle SW0 when mounting is completed.
MT PX TY Label (X and Y octal digits; a list of such messages is followed by a console switch loop.)	* Mount the indicated tape on channel X, tape Y. Toggle SW0 when mounting is completed.
Conventional magnetic tape messages	Follow procedures described in Symbolic Input/Output System (CD225E2.001).
TOGGLE SW0 TO CONTINUE	* (This message is typed each time a merge pass is completed in a multipass variable merge.) Toggle SW0 to continue; the program can be removed at this point for later completion if necessary.
END MERGE	The merge object program run is completed. Toggle SW0 to load the next card program (reads one binary card into 0 and branches to 0).

*Variable merges only

XI. BRIDGE II COMPATIBLE (CD225G1.006)

GENERAL DESCRIPTION

The user who has an 8k computer memory finds considerable advantage in using the BRIDGE II Compatible version of the FORWARD Sort/Merge Generator. This is particularly true of the user who is already maintaining his system tape with the BRIDGE II Operating System.

The CD225G1.006 generator is released as a system tape built by BRIDGE II. It can be updated under BRIDGE II control and can be added to a tape with other systems. For directions in this procedure, see BRIDGE II Operating Service System, CD225J1.001.

The CD225G1.006 version of FORWARD also has the option of generating an object program which can be executed from a BRIDGE II-built tape in sequence with other runs. When the user chooses this option, FORWARD generates a program which is referred to as the "tape object program."

Tape Object Program. When the user specifies data in columns 12 and 13 of the program parameter card, an object program is generated which may be executed either from cards or under BRIDGE II control from tape. The decision can be implemented from the tape list card. The procedure for executing the tape object program is to assemble the FORWARD symbolic output (the call for assembly is automatic with generator CD225G1.006 when switch zero is toggled), then load the assembled program (cards) to tape with BRIDGE II along with other runs. If columns 12 and 13 of the program parameter card are left blank, the object program can only be executed from the card reader.

The symbolic output of CD225G1.006 is always in the binary recording mode. The BRIDGE II Compatible version of the General Assembly Program must be used for assembly. (The CD225G1.002 generator produces symbolic output in the binary coded decimal mode.)

When a tape object program is executed under BRIDGE II control, the sort/merge program gives control to the run locator by way of a standard run completion routine upon conclusion of processing. The run completion routine is generated automatically for all tape object programs.

When a priority control channel and tape number is specified as the object program medium, the FORWARD Generator prepares the object program to be run from the indicated tape and channel. This means that a sort object program expects to read the rotary-merge subprogram (which is an overlay) from the specified tape. A sort or merge object program concludes its processing by searching the specified tape for a run locator.

If the name of the next object program is given on the program card, the run locator automatically locates it after the execution of the sort or merge object program. If no "next program name"

is specified, the run locator reads a control card from the card reader. When the sort or merge object program is executed, the operator may choose to cause it to operate as a card program by using an option of the tape list card, or he may change the assumed instruction tape priority control channel and tape number which has been specified at generation time. Thus, an object program generated to be run from an instruction tape may be run either from tape or cards, but a card program may be run only from cards.

FORWARD does not actually place the object program on the specified tape handler after generation. It generates a program compatible with BRIDGE II conventions (overlay labels, run completion routine, etc.) so the object program can be operated from a tape.

Columns 12-13 of the program card are reserved to indicate the instruction tape address. When a tape address is specified, the user should fill in the program card columns 41-49 either with the name of the next run to be executed or with the literal "NEX$\Delta\Delta\Delta\Delta\Delta$" for conventional run locator action in the object program. Columns 12-13 should either be blank or should contain a legitimate magnetic tape channel and tape number. Columns 41-49 should be blank whenever columns 12-13 are blank, but should otherwise contain linkage information as specified above.

The run completion routine uses a read console halt loop before executing next program search. By placing switch 19 down, next program control can be changed using a schedule control card.

Conventions

1. When the output of the General Assembly Program is a binary card deck, it will be compatible with the BRIDGE II card collection conventions.

2. If the output of the General Assembly Program is tape, the user must insure that this tape is compatible with the BRIDGE II tape collection conventions.

3. The General Absolute Loader (GAL) CD225B1.013R is recommended for loading absolute programs.

OPERATING CONSIDERATIONS

The run completion routine in the FORWARD Sort/Merge Generator can take advantage of one systems tape containing both FORWARD Sort/Merge and General Assembly Program II (CD225F1.008). However, this is not a requirement. Upon completion of generation, the symbolic program is on tape handler 3 and control is passed to the run locator. The run locator indicates it is ready to search for the General Assembly Program. At this time, control can be taken by the operator (and tape 3 may be assembled later), control can be given back to the run locator with no change (toggle switch zero), or control can be given back to the run locator modified by information contained on an SCC card which is the next card in the card reader (place switch 19 down and toggle zero). This last choice could be used to execute another job or call in the assembly system from another tape.

To execute the FORWARD Sort/Merge system, the following tape assignments are necessary. All tapes must be assigned to the same priority control channel.

FORWARD system tape	Number is optional
Blank	Handler 3
Blank	Handler 4
*Blank	Handler 5

*When assembly immediately follows the generation.

If the instruction tape used to execute a tape object program is created by BRIDGE II, forming a single card image per magnetic tape block, no special arrangements in the sort or merge are required for object loading. Otherwise, the user should reserve a #CA area in the input or output parameter card which is large enough to account for the actual block size. The #CA area is then used during loading of the object program, but is free thereafter for the use of ICE or OCE. The user must insure that his object loader uses #CA as an input area in this case. Assuming the instruction tape contains k card images per block ($k \leq 5$), the reserved area should be 40k + 2 words. As was stated above, no provision need be made if k = 1. A blocking factor of one is recommended so the General Absolute Loader (GAL) may be used "as is."

Special options are provided at object program execution time on the tape list card for tape object programs. To change the instruction tape address specified at generation time, the user punches the new channel and tape numbers in columns 79-80 of the tape list card. To operate the tape object program as a card program, the user punches "CD" columns 79-80 of the tape list card, and if the object program is a sort, he removes the overlay label which is the first card following the presort transfer card. When the tape list card is used for either of these purposes, the user may either supply a new tape list in columns 1-24 or he may leave columns 1-24 blank and punch only columns 79-80, in which case the original tape list is used. Whenever a tape list card is used with a tape object program, however, columns 79-80 must contain either "CD" or the instruction tape channel and tape number. For further information, refer to Chapters IX and X.

To generate a program from a system tape under control of BRIDGE II, the parameter deck is preceded by a sequence control card. A BRIDGE II start card is also necessary if the generation is the first run of a sequence or the only run.

The run label on the FORWARD system tape is "RUNFORWARD △△". The assumed run label of the General Assembly Program that is called is "RUNGAPII △PO△". The tape label of the CD225G1.006 version of FORWARD is "SYSTEM △BR".

Tape Object Program Messages

When a tape object program is run without a tape list card, it produces the same messages as when the object program is assembled to be run from cards. When the tape object program is run with a tape list card, the object program produces one of the two messages CARD OPTION or TAPE OPTION INST TAPE PXTY after the message TAPES OK, NO.

Generation Messages

Upon conclusion of generation and before assembly by use of the General Assembly Program, the following typeouts occur:

```
FORWARD GENERATOR
TEST PGM SYMBOLIC ON P1  T3

RUNFORWARD   COMPLETED
LOCATING RUNGAPII PO
```

The program halts to allow the operator time to set switch 16 and toggle switch 0 to continue.

XII. DEFINITION OF TERMS

Definitions of most terms used in this publication may be found in CPB-178, GE-225 Programming Conventions. In addition, the following terms appear in this publication.

BLOCK - used to denote "physical tape record" to avoid confusion. A block is understood to contain an integral number of complete records.

COLLATION TAPE - a tape onto which strings are dispersed and from which strings are merged or collated in a sort.

COLLATION TAPE LIST - a list of all the collation tape handlers used in a sort.

DISPERSION - writing of successive strings alternately on a set of two or more tape handlers.

LEVEL - in a merge combining more files than there are tape handlers, a level is any part of the merge in which a set of files is combined into a single file in one movement of data through the central processor.

MERGE ORDER - the number of input files which are collated by a merge concurrently.

MULTILEVEL MERGE - a merge program requiring several levels to combine all of the input files.

RECORD - used to denote "logical record" rather than "physical tape record."

SEQUENCE KEY - a set of characters within a record, having a logical order of significance and used to identify the record and to determine its place in a file or in an ordered sequence of records.

SORT - a program which rearranges the records of a file into sequence on an assigned key.

STRING - a set of records arranged in ascending sequence according to increasing magnitude of their keys.

STRING BREAK - the point at which a sort can find no more records with sufficiently high sequence keys to fit on the current output string.

STRING LENGTH - the number of records in a string.

TOURNAMENT - a sequence key comparison procedure which minimizes the number of comparison steps necessary to determine the least key record among those being processed.

APPENDIX A. MULTIREEL INPUT FILES

Section II recommends sorting of only a single full reel of data at a time. It is not the multiple input reels which cause difficulty, but multiple collation tapes. When the output collation tape becomes full during any intermediate pass of a sort, it is necessary to remove it and replace it with a blank. Since each intermediate output tape is subsequently used as input in a later collation pass, tape changing is again necessary when the tapes are required for input. Generally speaking, if enough records enter the sort to require tape changing in one pass, tape changing is necessary in all passes -- not merely on the output handler, but on the input handler as well. Thus, the recommended limit of one full reel of data at a time means one full collation reel. It is impractical to risk tape changing in a sort involving large amounts of data in order to save a post-sort merge. The dubious convenience of avoiding the merge is more than offset by operational inconvenience in the sort, and usually by sharply increased sort time resulting from the rewinds for tape changing.

Because the collation tape blocking factor is made as large as possible in every sort, collation tapes can usually hold more logical records than can ordinary input or output files with smaller blocking factors. (When the single pass option is used, however, the collation tape blocking factor is the same as the output blocking factor.) It is often possible to sort a greater number of logical records than contained in a single full input reel without exceeding a full reel of any collation tape. By use of the tables which follow, one can predict the number of records which can enter a sort without causing collation tape swapping. The table values indicate how many input reels may enter the sort. Each column is headed by an assumed number of reels of input. The row headings specify the tape storage mode. Entries in the body of the table indicate the maximum physical record (block) size -- the number of words of data per block which may be used on the input to permit the given number of reels to be handled without tape swapping. The derivations of the formulas are given in later paragraphs.

Maximum Number of Words in Input Block--4k Memory

The tables assume 6 or fewer collation tapes and no more than 400 words used for OCE + SCE + #CS + #CA.

1. Input and collation tape densities of 200 bits per inch.

	Reels of Input		
Tape Storage Mode	1.5	2	3
BCD or 18-Bit Binary	53 Words	29 Words	17 Words
20-Bit Binary	43 Words	25 Words	14 Words

2. Input density 200 bits per inch and collation tape density 556 bits per inch.

Tape Storage Mode	Reels of Input 1.5	2	3	4	5
BCD or 18-Bit Binary	Any *	175	43	29	21
20-Bit Binary	Any *	175	43	25	19

3. Input density and collation tape density 556 bits per inch.

Tape Storage Mode	Reels of Input 1.5	2	3	4
BCD or 18-Bit Binary	79	53	29	21
20-Bit Binary	72	43	25	19

* The indicated number of reels may be sorted at once regardless of input block size, but for other reasons it is recommended that the input block size not exceed 175 words.

Maximum Number of Words in Input Block--8k Memory

The tables assume 6 or fewer collation tapes and not more than 400 words used for OCE + SCE + #CS + #CA.

1. Input and collation tape densities of 200 bits per inch.

Tape Storage Mode	Reels of Input 1.5	2	
BCD or 18-Bit Binary	76 Words	41 Words	21 Words
20-Bit Binary	60 Words	31 Words	16 Words

2. Input density 200 bits per inch, collation tape density 556 bits per inch.

Tape Storage Mode	Reels of Input 1.5	2	3	4	5	6	7
BCD or 18-Bit Binary	Any **	Any **	125	55	38	27	21
20-Bit Binary	Any **	Any **	100	50	31	22	17

3. Input density and collation tape density 556 bits per inch.

Tape Storage Mode	Reels of Input					
	1.5	2	3	4	5	6
BCD or 18-Bit Binary	142	83	45	33	25	20
20-Bit Binary	125	71	38	26	20	16

* The indicated number of reels may be sorted at once regardless of input block size, but for other reasons, it is recommended that the input block size not exceed 500 words.

Derivation of Tables

The tables were computed under the assumptions that:

1. Six or fewer collation tapes are used.
2. All tapes are of standard 2400-foot lengths.
3. No more than 400 words in a 4k memory nor 600 words in an 8k or 16k memory are used for OCE + SCE + #CA + #CS.

If the above assumptions are not satisfied, a block size smaller than indicated in the tables should be used.

As an example of the use of the tables, consider BCD card image records (27 BCD words), one per block, with 200 bits per inch density on all tapes (15kc). The tables indicate that two reels may be sorted at once in either 4k or 8k memories, since 27 is less than 29 in table 1 for 4k memories and less than 41 in table 1 for 8k memories.

Often it is impossible to satisfy the limits stipulated in the tables, because of input blocks which are too large. In such circumstances, the best rule of thumb is to enter one reel of input at a time. The collation tape block sizes almost always exceed input block sizes, so that one reel of input will almost never cause collation tape swapping. (The Single Pass Option exception has already been noted.)

There is a safety factor built into the table values. Thus, if it is known that the input data consists of no more than, say, 1.2 full reels, and the input block size is not over 20 percent larger than the table value, it is usually safe to permit all of the data to enter the sort at once. If the input consists of several small files which together do not exceed 1.2 full reels, the same considerations apply. The derivations of the formulas used to compute table values are now given.

In order to determine how many logical records a collation tape can hold, it is necessary first to compute the sort blocking factor. The formula specifies that the sort blocking factor will be the largest integer not exceeding:

$$(A) \qquad \frac{\text{memory size} - 1550 - \text{OCE} - \text{SCE} - \#\text{CA area} - \#\text{CS area}}{2 \times \text{record size} \times \text{number of collation tapes}}$$

For example, if an 8192-word memory and 5 collation tapes are used to sort a file of 27-word logical records, with OCE etc., requiring 300 words:

$$\text{blocking factor} = \frac{8192 - 1500 - 300}{2 \times 27 \times 5} = \frac{6392}{270} = 24$$

When the blocking factor is known, the number of logical records stored on a full reel of tape is approximately equal to:

$$\frac{\text{(tape length in inches)}}{\left[\text{(block length in inches)} + \text{(inter-record gap)}\right] \text{(blocking factor)}}$$

which is:

$$\left[\frac{(12)\ \text{(tape length in feet)}\ \text{(blocking factor)}}{\text{(blocking factor)}\ \text{(record size in characters)}} + (0.75)\right]$$
$$\text{(recording density in bits per inch)}$$

For 2400 foot reels of tape, this reduces to:

(B) $$\frac{(28800)\ \text{(density in bits per inch)}\ \text{(blocking factor)}}{\text{(blocking factor)}\ \text{(record size in characters)} + (.75)\ \text{(density)}}$$

This formula may be used for any fixed record length tape, whether a sort is involved or not.

The criterion for the amount of data that should be allowed to enter the sort at one time is now established. It is the number of records that can be held in a single collation tape, as blocked by the sort program. The formulas A and B are adequate to compute this limit.

However, it is often much more practical to inquire how many full reels of input should be sorted at a time, rather than how many logical records. (The tables are based on number of full reels of data.) The limit is:

$$\text{no. of input reels} = \frac{\text{number of logical records a collation tape can hold}}{\text{number of logical records each input tape holds}}$$

Applying the formulas discussed above, this ratio reduces to:

(C) $$\frac{\text{(coll. density)}\ \text{(coll. blk. factor)}\ .\left[\text{(inp. blk. factor)}\ \text{(rec. size in char.)} + .75\ \text{(inp. den.)}\right]}{\text{(inp. den.)}\ \text{(inp. blk. factor)}\ .\left[\text{(coll. blk. factor)}\ \text{(rec. size in char.)} + .75\ \text{(coll. den.)}\right]}$$

In the previous example, it was estimated that the collation tape blocking factor would be 24 for 27-word logical records with 5 collation tapes and an 8192-word memory. Assuming that the data being sorted are BCD records blocked singly on 200 bit-per-inch tapes, with 200 bit-per-inch collation tape density also used, the limit using formula C is:

$$\frac{200)\ (24)\ .\left[(1)\ (81) + 150\right]}{(200)\ (1)\ .\left[(24)\ (81) + 150\right]} = 2.6 \text{ reels}$$

Thus in this example 2.6 reels of input may be sorted at once without incurring collation tape changing. In practical circumstances, the estimate would be truncated to 2 reels which is the same as given in the tables. (The file described typifies many card-to-tape files.)

In summary, the table values are based on the following inequality - if:

1. Q is the number of input reels that may be conveniently sorted at once.
2. d_c and b_c are the collation tape density and blocking factor.
3. d_i and b_i are the input tape density and blocking factor.
4. r is the number of characters (on tape) in each logical record.

then:

$$Q \leq \frac{b_c\ d_c\ (0.75\ d_i + b_i\ r)}{b_i\ d_i\ (0.75\ d_c + b_c\ r)}$$

Solving the inequality for $b_i\ r$, which is the number of characters in each input block, gives:

$$b_i r \leq \frac{0.75\ d_i\ d_c\ (b_c - Q\ b_i)}{b_c\ (Q\ d_i - d_c)}$$

The inequality may be simplified when numeric values for densities are substituted:

1. For $d_i = d_c = 200$, $\quad b_i\ r \leq \dfrac{150\ (b_c - Q\ b_i)}{(Q-1)\ b_c}$

2. For $d_i = 200$ and $d_c = 556$, $\quad b_i\ r \leq \dfrac{83400\ (b_c - Q\ b_i)}{(200\ Q - 556)\ b_c}$

3. For $d_i = d_c = 556$, $\quad b_i\ r \leq \dfrac{417\ (b_c - Q\ b_i)}{(Q-1)\ b_c}$

Next, applying the assumption that $b_c\ r = k\ b_i\ r$, the inequalities limiting $b_i\ r$ become more manageable: ($b_c\ r$ is the number of characters in each collation tape block.)

(D) $\quad b_i\ r \leq \dfrac{150\ (b_c - Q\ b_i)}{(Q-1)\ b_c} \quad$ becomes $\quad k \geq \dfrac{(Q-1)\ b_c\ r + Q}{150}$

For convenience, it was assumed that $b_c\ r = k\ b_i\ r$. A reasonable (constant) value for $b_c\ r$ was computed and used throughout. For 4k memories, with 6 or fewer tapes and no more than 400 words used by OCE, etc., the collation tape block size may be expected to be about 175 words; for 8k memories, with 6 or fewer tapes and no more than 600 words of OCE., the collation tape block size may be expected to be about 500 words. In the table computations, the collation tape block size was assumed to be $b_c\ r = 525$ characters for BCD tapes and 4k memory, $b_c\ r = 700$ characters for 20-bit binary tapes and 4k memory, $b_c\ r = 1500$ characters for BCD tapes and 8k memory, and $b_c\ r = 2000$ characters for 20-bit binary tapes and 8k memory.

(E) $\quad b_i\ r \leq \dfrac{83400\ (b_c - Q\ b_i)}{(200\ Q - 556)\ b_c} \quad$ becomes $\quad k \geq \dfrac{(200\ Q - 556)\ b_c\ r + Q}{83400}$

(F) $b_i\ r \leq \frac{417\ (b_c - Q\ b_i)}{(Q-1)\ b_c}$ becomes $k \geq \frac{(Q-1)\ b_c\ r + Q}{417}$

Here k may be thought of as roughly equal to the number of input blocks required to make one collation tape block.

The computations were done in two steps:

With Q and $b_c\ r$ given, k was computed, using formulas D, E, and F then $b_i\ r \leq \frac{b_c\ r}{k}$ was computed.

Rounding or truncation on intermediate results was always in favor of conservative conclusions, so that the table values would be "safe" in terms of avoiding collation tape swapping.