HONEYWELL

SOFTWARE

MOD 1 (MSR)

UTILITY ROUTINES

GENERAL INFORMATION:	SERIES 200/OPERATING SYSTEM – MOD 1 (MASS STORAGE RESIDENT)
SUBJECT:	Programming and Operating Procedures for the Utility Routines of the Mod 1 (MSR) Operating System.
SPECIAL INSTRUCTIONS:	This edition completely supersedes the manual of the same name dated May 10, 1968. It is one of a series of manuals describing the Mod 1 (MSR) Operating System. Refer to the preface for other related documentation. The portions of this publication containing new and changed information are indicated on page iii.

INCLUDES UPDATE PAGES PUBLISHED AS ADDENDA NO. 1 ON AUGUST 6, 1969, NO. 2 ON JANUARY 5, 1970, AND NO. 3 ON JULY 15, 1970.

DATE: November 7, 1968

FILE NO.: 123.6905.141C.3-<u>619</u>*

0976
12.5M
5570
Printed in U. S. A.

*Underscoring denotes Order Number.

PREFACE

This manual describes the utility routines for the Series 200/Operating System - Mod 1 (Mass Storage Resident). These routines provide facilities for: (1) volume initialization, (2) data sorting, (3) dumping, (4) disk/tape copy, and (5) volume modification for the Mod 1 (MSR) Operating System. Besides this manual, other pertinent publications include the following:

Series 200/Mod 1 (MSR) Operating System Summary Description (Order No. 615);

Supervisor (Order No. 616);

Program Development Subsystem (Order No. 617);

Data Management Subsystem (Order No. 618); and

Operating Procedures (Order No. 620).

The Summary Description cited above is prerequisite reading to this manual as well as to the other manuals listed. In addition, a publications guide is provided in the Summary Description to aid the reader in his study of the system.

Section I of this manual provides a brief description of each of the utility routines in the Mod 1 (MSR) Operating System. Section II describes Volume Preparation C; Section III describes Bootstrap Generator C; Section IV describes Mass Storage Sort C; Section V describes Mass Storage Edit C; Section VI describes Disk/Tape Copy C; and Section VII describes Volume Modification D. Appendix A describes the method of calculating sort-item block size. Appendix B describes general procedures for handling bad tracks on a mass storage volume and procedures specific to Volume Preparation C, Mass Storage Sort C, and Disk/Tape Copy C.

The Mod 1 (MSR) Utility Routines are a set of coded programs designed to extend the power of Series 200 in the area of utility functions. They are supported by comprehensive documentation and training; periodic program maintenance and, where feasible, improvements are furnished for the current versions of the programs, provided they are not modified by the user.

NEW AND CHANGED INFORMATION

This edition has been revised to reflect extensions to the Mod 1 (MSR) Operating System. Section VII has been added to describe a new routine, Volume Modification D. Volume Preparation C has been augmented to include several new features, including the ability to provide substitute tracks for bad tracks on the mass storage volume being prepared. Disk/Tape Copy C now includes two copy methods, copy-by-rectangle and copy-named-files, which can be used with all copy options. Appendix B has been added to describe general procedures for handling bad tracks on a mass storage volume and procedures specific to Volume Preparation C, Mass Storage Sort C, and Disk/Tape Copy C.

This edition contains no references to loading utility routines from cards.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I
INTRODUCTION

The utility routines of the Series 200/Operating System — Mod 1 (Mass Storage Resident) provide facilities for: (1) volume initialization, (2) data sorting, (3) dumping, (4) disk/tape copy, and (5) volume modification for mass storage. These utility routines are Volume Preparation C, Bootstrap Generator C, Mass Storage Sort C, Mass Storage Edit C, Disk/Tape Copy C, and Volume Modification D.

VOLUME PREPARATION C

Volume Preparation C prepares a mass storage volume for use under the data management conventions of the operating system. This routine formats tracks of the volume while checking for bad surface areas, writes the volume label, and creates the volume directory. The routine can be directed to provide substitute tracks for bad tracks on the volume. Volume Preparation C must be performed once for every mass storage volume at the time the volume is first entered into the operating system.

BOOTSTRAP GENERATOR C

Bootstrap Generator C creates a bootstrap routine on a mass storage volume. Bootstrap Generator C can also re-create a bootstrap routine at any time after its initial creation and specialize it for the requirements of a specific installation. The bootstrap routine is used to bring the Supervisor into main memory from mass storage.

MASS STORAGE SORT C

Mass Storage Sort C can sort items from a mass storage, card, or tape file, using work areas on a mass storage volume. The principal operation is a key sort, in which the sort key fields are taken from source items and only the keys are sorted. The Fetch macro routine is used to retrieve the sorted keys and the associated source items. If desired, selected fields can be extracted from each source item and carried through the sort process along with the key. The output of the sort then contains the extracted fields as well as the sort keys. At this point, the Fetch macro routine is used to retrieve these extract fields from the sorted output, thus eliminating the necessity of retrieving them from the original source items.

A further option of Mass Storage Sort C permits the entire source item to be sorted. This method of sorting eliminates the need to retrieve fields from the source items.

The sequential output file program can be used to create a sequential mass storage file (as described in the Mod 1 (MSR) Data Management Subsystem manual) or a tape file of the source items in the sorted output sequence.

MASS STORAGE EDIT C

Mass Storage Edit C provides a printed representation of the physical data stored on a mass storage volume. Parameters to the edit routine specify the area(s) to be edited. The edit routine does not recognize the existence of files as such; therefore, a user wishing to produce a listing of a specific file should refer to the Mod 1 (MSR) Data Management Subsystem manual.

DISK/TAPE COPY C

Disk/Tape Copy C provides a fast method of copying rectangular areas or named files from a mass storage volume for backup purposes. Disk/Tape Copy C copies up to 16 rectangular areas or any number of named files from a mass storage volume onto another mass storage volume or onto magnetic tape. The copy on magnetic tape may subsequently be written back, in whole or in part, onto a mass storage volume by this same routine.

VOLUME MODIFICATION D

Volume Modification D is used to create the track substitution capability on a mass storage volume, if this capability was not previously created by Volume Preparation C. After the track substitution capability is created, Volume Modification D can be used to provide substitute tracks for known bad tracks that were not provided with substitutes during Volume Preparation C. At the end of the run, Volume Modification D produces a printer listing of all bad tracks for which substitute tracks have been provided and all substitute tracks on the volume.

UTILITY ROUTINES IN A MULTIPROGRAMMING ENVIRONMENT

Volume Preparation C, Mass Storage Edit C, and Volume Modification C cannot be run in a multiprogramming environment. Other utility routines of the Mod 1 (MSR) Operating System can be run in a multiprogramming environment, but only as background programs.

Assignments of read/write channels are made automatically. When a utility routine operates in a single-job environment, it makes optimum use of all available read/write channels. When a utility routine is run as a background program in a multiprogramming environment, it does not use read/write channel 1: that channel is reserved exclusively for the foreground program.

SECTION II
VOLUME PREPARATION C

Volume Preparation C is a utility routine of the Mod 1 (MSR) Operating System. The routine prepares a mass storage volume for use under the system's data management conventions. (For a description of these conventions, see the Data Management Subsystem manual.)

Volume Preparation C must be run for each mass storage volume as it is first entered into the operating system. Thereafter, whenever reformatting of the entire volume is desired, Volume Preparation C must be run again.

Volume Preparation C cannot be run in a multiprogramming environment.

EQUIPMENT REQUIREMENTS FOR VOLUME PREPARATION C

Basic Equipment Requirements

The following basic equipment is required:

A Series 200 central processor;

Advanced Programming Instructions;

12,288 characters of main memory; and

One disk device and associated control.

This combination may be any one of the following.

Device Type	Control Type
155	157C, 257C
258	257, 257-1, 260
259	257, 257-1, 260
273	257, 257-1, 260
259A[1]	257A
259B[1]	257B
261	260
262	260

[1]When a mass storage volume is mounted on a Type 259A or 259B Disk Pack Drive, the value 259 must be used in the device-type parameter. Omitting the device-type parameter produces the same effect in this case, since 259 is the assumed value if no device type is specified.

Also required are one card reader and one printer.

Additional Usable Equipment

The following additional equipment may be used:

One Type 220-1, -3 Console (console I/O requires 4,096 additional characters of main memory);

Second I/O Sector (Feature 1115 for Model 2200); and

One or more additional disk devices as tabulated above.

FUNCTIONAL DESCRIPTION OF VOLUME PREPARATION C

Volume Preparation C performs the following functions:

1. Writes the volume label,
2. Creates the volume directory, and
3. Checks for bad disk surface areas[1] by formatting either the entire mass storage volume or only certain critical tracks (see the "Surface-Check Parameter" later in this section).

When Volume Preparation C formats the entire mass storage volume, each track (except for those tracks reserved for system use on cylinder 000) is formatted so that it consists of equal length physical records with no track-linking records.

The length of the physical records on each track is determined by the device type specified (or defaulted) in the device-type parameter of the job control file. The record lengths are as follows:

Device Type	Length of Record
155, 258, 259, 273	271 characters
261, 262	256 characters

After being formatted, each track is read back. If an error occurs that cannot be corrected by reformatting, the routine uses a bad track handling procedure.[1]

JOB CONTROL LANGUAGE FOR VOLUME PREPARATION C

In order to execute Volume Preparation C, a job control file, consisting of a series of job control statements, must be placed in the card reader. Each job control statement contains one or more parameters.

In the following description of job control statements, the term "line" is the equivalent of one card in the job control file.

[1] "Bad Track Handling Procedures for Volume Preparation C" appears in Appendix B of this manual.

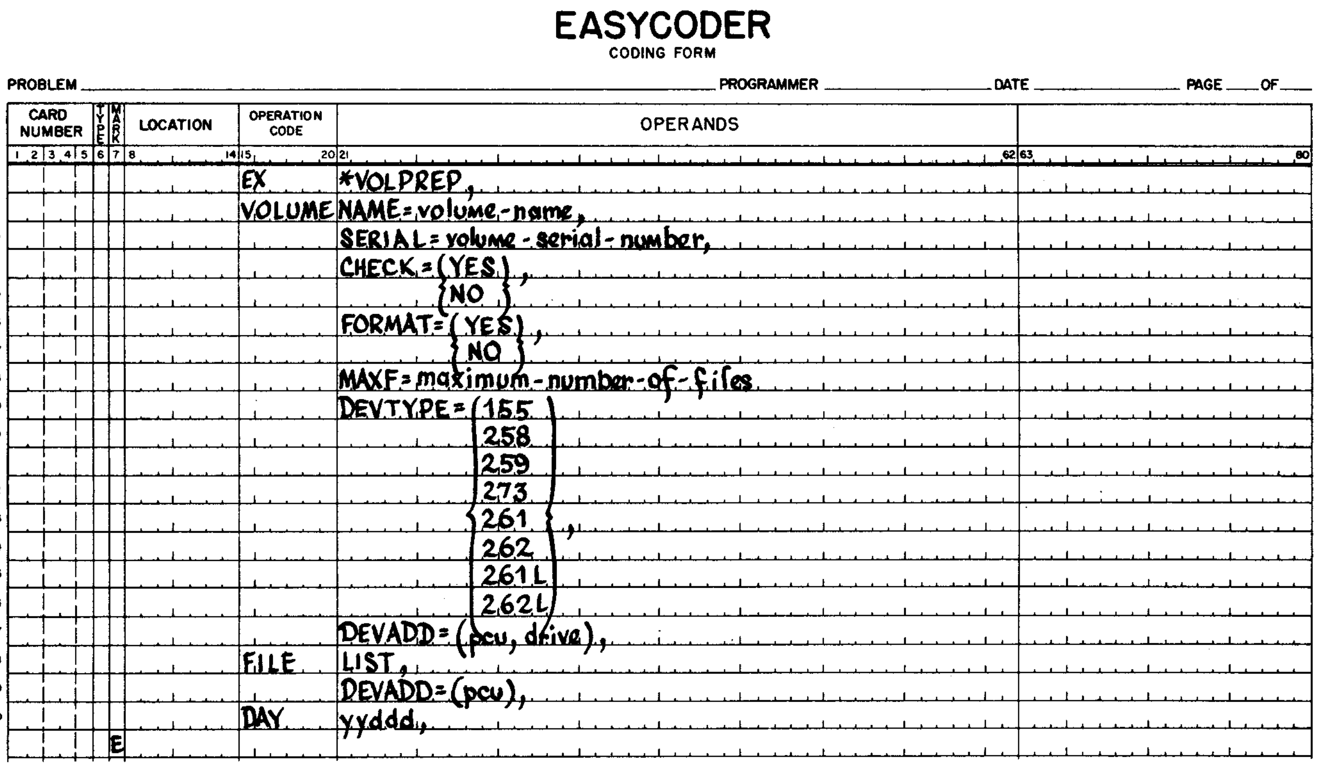
Sequence of Job Control Statements

Job control statements for Volume Preparation C must appear in the following order:[1]

1. Execute statement (if the routine is to be loaded from mass storage),
2. Volume statement,
3. File statement (if present), and
4. Day statement (if present).

The last line of the last job control statement must contain an E in its mark field or be followed by a line with blanks in its operation code and operands fields and an E in its mark field.

Figure 2-1 shows the basic set of job control statements for Volume Preparation C.[1]



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER | TYPE | MARK | LOCATION | OPERATION CODE | OPERANDS

```
              EX    *VOLPREP,
              VOLUMENAME=volume-name,
                    SERIAL=volume-serial-number,
                    CHECK={YES},
                          {NO }
                    FORMAT={YES},
                           {NO }
                    MAXF=maximum-number-of-files
                    DEVTYPE={155 }
                            {258 }
                            {259 }
                            {273 }
                            {261 },
                            {262 }
                            {261L}
                            {262L}
                    DEVADD=(pcu, drive),
              FILE  LIST,
                    DEVADD=(pcu),
              DAY   yyddd,
      E
```

Figure 2-1. Job Control Statements for Volume Preparation C

Execute Statement

The Execute statement is used only when Volume Preparation C is to be loaded from mass storage. The Execute statement directs the Supervisor to load Volume Preparation C from the system disk, which is mounted on logical drive number 0. An additional mass storage device, whose drive number can be from 1 through 7, is required for the volume being prepared (unless the system residence volume is being prepared). An Execute statement is required for every run of this routine when it is mass storage resident.

[1]See Appendix B for additional job control language elements that apply to bad track handling.

The Execute statement is not used when Volume Preparation C is loaded from magnetic tape. (A drive number from 0 through 7 can be assigned to the mass storage device holding the volume being prepared.) For a description of loading from magnetic tape, see "Operating Procedures for Volume Preparation C," later in this section.

Volume Statement

The Volume statement is required for every run of Volume Preparation C because the name of the volume to be prepared must always be specified in the volume-name parameter. All other parameters of the Volume statement are optional; if these parameters are omitted, their default values are used during processing.

VOLUME-NAME PARAMETER

The volume-name parameter (NAME) is required for every run of Volume Preparation C. The volume-name parameter provides the name to be written onto the volume label. The form of this parameter is shown below.

$\boxed{\text{NAME = volume-name,}}$

volume-name =	This name can be from one through six alphanumeric characters in length. Letters A through Z and numbers 0 through 9 can be used. The last character must be directly followed by a comma. If fewer than six characters are specified, this field on the volume label is zero-filled to the left.

VOLUME-SERIAL-NUMBER PARAMETER

This parameter allows the user to specify an alphanumeric value, up to six characters in length, which supplements the volume name in identifying the mass storage volume. Like the volume name, the volume serial number is written onto the volume label. The form of this parameter is shown below.

$\boxed{\text{SERIAL = volume-serial-number,}}$

volume-serial-number =	This value may be up to six alphanumeric characters in length. If fewer than six characters are specified, this field on the volume label is zero-filled to the left.

Default assumption: if this parameter is omitted, the volume name is also written into the volume-serial-number field on the volume label.

VOLUME-SERIAL-NUMBER-CHECK PARAMETER

This parameter enables the user to insure that the correct volume has been mounted. When this parameter contains the keyword value YES (or when the parameter is omitted), the volume-serial-number field on the volume label is checked for equality with the value specified in the volume-serial-number parameter. In this case, the volume-serial-number parameter must be specified. The form of this parameter is shown below.

$$\boxed{\text{CHECK} = \left\{\begin{matrix}\text{YES}\\ \text{NO}\end{matrix}\right\} ,}$$

YES = The volume-serial-number field of the volume label is checked for equality with the value specified in the volume-serial-number parameter. If inequality results, a halt or console message occurs.

NO = The above check is not performed. If the volume-serial-number parameter is specified, its value is written into the volume-serial-number field on the volume label. If the volume-serial-number parameter is not specified, the volume name is written into the volume-serial-number field on the volume label.

Default assumption: YES

SURFACE-CHECK PARAMETER

This parameter enables the user to specify whether formatting and surface checking are to be performed for the entire mass storage volume or only for certain critical tracks. The form of this parameter is shown below.

$$\boxed{\text{FORMAT} = \left\{\begin{matrix}\text{YES}\\ \text{NO}\end{matrix}\right\} ,}$$

YES = The entire mass storage volume is to be formatted and checked for bad tracks.

NO = Only the following areas are to be formatted and checked for bad tracks: the bootstrap track, the substitute bootstrap track, the volume label, the volume directory, the *BADTRACKS file (if created), and the *VOLSPARES file (if created).[1]

Default assumption: YES

MAXIMUM-NUMBER-OF-FILES PARAMETER

The maximum-number-of-files parameter designates the maximum number of files that are expected to be stored on the volume being prepared. Sufficient space is allowed in the volume directory to accommodate the number of files designated by this parameter.

The user may specify a number from 1 to 26 for a volume mounted on a Type 155 Disk Pack Drive or 1 to 86 for a volume mounted on a Type 258, 259, 273, 259A, or 259B Disk Pack Drive. A number from 1 to 158 may be specified for a volume mounted on a Type 261 or 262 Disk File. Volume Preparation C uses a standard value for the maximum number of files, based on the value specified by the user (see Table 2-1 and 2-2).

[1]The *BADTRACKS and *VOLSPARES files are described in Appendix B under "Bad Track Handling Procedures for Volume Preparation C."

The user must specify an adequate number of files since the only means of increasing the number of files allowed is to (1) unload all files, (2) run Volume Preparation C with a larger value in the maximum-number-of-files parameter, and (3) reallocate and reload all files.

The form of this parameter is shown below.

$\boxed{\text{MAXF = maximum-number-of-files,}}$

maximum-number-of-files = A decimal number from 1 to 26 for the Type 155 Disk Pack Drive or 1 to 86 for Type 258, 259, 273, 259A, and 259B Disk Pack Drives; from 1 to 158 for Type 261 and 262 Disk Files.

NOTE: The MAXF parameter card is not required for the Type 155 Disk Pack Drive as MAXF is always set to 26. Tracks reserved for the Volume Directory are cylinder 00, track 01, records 1 through 14, and cylinder 01, tracks 00 and 01.

The user-specified value produces a standard value used by Volume Preparation C as shown in the following tables.

Table 2-1. MAXF Values for Type 258, 259, 273, 259A, and 259B Disk Pack Drives

Value Specified by User	Value Used by Volume Preparation C	Tracks on Cylinder 000 Reserved for Volume Directory
None (default)	26	2 - 4
1 - 26	26	2 - 4
27 - 56	56	2 - 6
57 - 86	86	2 - 8
Greater than 86	Invalid parameter value halt or console message occurs.	

Table 2-2. MAXF Values for Type 261 and 262 Disk Files

Value Specified by User	Value Used by Volume Preparation C	Tracks on Cylinder 000 Reserved for Volume Directory
None (default)	50	2 - 4
1 - 50	50	2 - 4
51 - 104	104	2 - 6
105 - 158	158	2 - 8
Greater than 158	Invalid parameter value halt or console message occurs.	

Default assumption: if this parameter is omitted, the value 26 is used for a volume mounted on a Type 155, 258, 259, 273, 259A, or 259B Disk Pack Drive: the value 50 is used for a Type 261 or 262 Disk File.

DEVICE-TYPE PARAMETER

The device-type parameter (DEVTYPE) specifies the type of device that holds the volume to be prepared. The form of this parameter is shown below.

$$\text{DEVTYPE} = \left\{\begin{array}{l}155\\258\\259\\273\\261\\262\\261\text{L}\\262\text{L}\end{array}\right\},$$

If the volume to be prepared is mounted on a Type 259A or 259B Disk Pack Drive, the value 259 must be used in the device-type parameter. Omitting the parameter produces the same effect in this case, since the default value is 259.

If a Type 262 Disk File is specified in this parameter, Volume Preparation C prepares only the one volume whose address is specified (or defaulted) in the device-address parameter.

Default assumption: if this parameter is omitted, the value 259 is used.

DEVICE-ADDRESS PARAMETER

The device-address parameter (DEVADD) indicates the address of the disk device holding the volume to be prepared. The form of this parameter is shown below.

$$\boxed{\text{DEVADD = (pcu, drive),}}$$

pcu	=	The peripheral control unit number, written as two octal characters. The I/O bit is not used, but all other bits, including sector bits, must be specified.
drive	=	The drive number, written as one octal character.

Default assumption: if this parameter is omitted, the default assumption is pcu 04, drive 0.

File Statement for the List File

The list file is a printer listing of bad tracks, if any, and their substitutes on the volume being prepared. The list file is described in Appendix B of this manual under "Bad Track Handling Procedures for Volume Preparation C."

The File statement for the list file is required only when the printer is assigned a peripheral control address other than 02. If this statement is omitted, the printer's peripheral control address is assumed to be 02.

The form of this statement is shown in lines 14 and 15 of Figure 2-1.

DEVICE-ADDRESS PARAMETER

This parameter is required when the File statement for the list file is used. This parameter specifies the address of the printer's peripheral control unit. The form of this parameter is shown below.

$$\boxed{\text{DEVADD = (pcu),}}$$

<u>pcu</u>	=	The peripheral control unit number, written as two octal characters. The I/O bit is not used, but all other bits, including the sector bits, must be specified.

<u>Day Statement</u>

The Day statement is optional. It is used to specify the creation date of the system files created by Volume Preparation C. When specified, the date is entered into the file description index item for each system file created by Volume Preparation C.

The form of this statement is shown on line 16 of Figure 2-1. The "yy" portion of the statement gives the year of the creation in decimal digits; the "ddd" portion gives the day of the creation in decimal digits (January 1st being day 001).

Default assumption: if this statement is omitted, the creation date is taken from the current date field of the Supervisor.

<u>OPERATING PROCEDURES FOR VOLUME PREPARATION C</u>

<u>Loading Volume Preparation C</u>

Volume Preparation C can be loaded in two ways.

1. From mass storage, by means of the Supervisor, under the Mod 1 (MSR) Operating System;
2. From magnetic tape, by means of Floating Tape Loader-Monitor C, under the Mod 1 (TR) Operating System.

MASS STORAGE RESIDENT

When Volume Preparation C is loaded from mass storage, an Execute statement (as illustrated on line 1 of Figure 2-1) must be submitted in the job control file, followed by the desired job control statements for the routine.

TAPE RESIDENT

Loading Volume Preparation C from magnetic tape is similar to loading from mass storage, except that a console call card is used instead of an Execute statement. The format of the console call card is shown in Table 2-3.

Table 2-3. Console Call Card Format

Characters	Contents	Meaning
1 - 8	*VOLPREP	Program and segment names for Volume Preparation C
9	d	Tape drive number from which Volume Preparation C is to be loaded
18	*	Asterisk (required by the Mod 1 (TR) Operating System)

Protection Switch Settings

Before running Volume Preparation C, the user must set to PERMIT all protection switches on the peripheral control unit of the disk device holding the volume to be prepared.

Operator Communication and Control

When an error condition occurs, 2-way communication between the operating system and the operator is achieved through a control panel or (if certain software and hardware requirements have been met) through a console. If the communication medium is a control panel, an error condition causes the central processor to halt. If the communication medium is a console, an error condition causes a message to appear, but the central processor does not halt.

Halts or console messages are caused by three types of error conditions: peripheral device conditions, job control file conditions, and conditions specific to Volume Preparation C. When a halt or console message occurs, the operator must determine which error condition exists and then make the appropriate response.

CONTROL PANEL HALTS

At the time of a control panel halt, the B-address register contains a value that identifies the error condition. In all cases except for card read errors, the A-address register contains the address of a response location in main memory. The response location is a 1-character main memory location into which the operator must enter a response character appropriate to the error involved. Where indicated, additional information concerning the error can be found in the main memory locations immediately following the response location. This additional information is called the supplementary list (see Table 2-4) and is composed of the following elements (arranged in ascending order of contiguous main memory locations).

Table 2-4. Supplementary List

Field Name	Length
Specific code[1]	One character (A+1)
Volume name	Six characters
Volume serial number	Six characters
Mass storage pcu address	One character
Device address	One character
Magazine number	One character (always 0)
Error address[1]	Six characters in the decimal format ccctt

[1]Significant only for certain mass storage peripheral device errors (otherwise blank).

Peripheral Device Halts

A B-address register value in the range from 0000 through 3777 indicates an error condition involving a peripheral device. The operator should consult Table 2-5. The B-address register value has the general form ppxd, where:

pp = Peripheral control unit (leftmost bit set to zero);

x = Code indicating type of error condition:

- 0 = Device not operable,
- 1 = Uncorrectable read error,
- 2 = Uncorrectable write error, and
- 4 = Positioning error;

d = Device number.

These values are consistent with the standard peripheral error halt codes.

When the halt condition involves the card reader, the A-address register does not contain meaningful values. In all other cases, the A-address register contains the response location. The supplementary list is also present (see Table 2-4).

Table 2-5. Peripheral Device Halts

B-Address Register Contents	Specific Code (A+1)	Condition	Operator Action[1]

pp0d	01
	02
		Device inoperable.
		Protection violation.
			G = retry.
			E = exit to the Supervisor.

pp1d		Read error (cards).	Correct erroneous card, if possible; refeed cards, starting with that card; and press RUN.

pp2d	04
	12
		Format violation.
		Track overflow.
			G = retry.
			E = exit to the Supervisor.

	10	Write error.[2]	G = retry.
			S = skip the bad track and resume processing.
			E = exit to the Supervisor.

pp4d	03	Positioning error.	G = retry.
			E = exit to the Supervisor.

	13	Positioning error.	E = exit to the Supervisor.

[1]The operator enters the appropriate response character into the response location and presses RUN (G = 27_8, S = 62_8, E = 25_8).

[2]If *BADTRACKS and *VOLSPARES files have not been created on the volume and a bad track is encountered during formatting (write error), the following message appears on the printer: CYL nnn TR nnn ERROR (nnn is a decimal value). If *BADTRACKS and *VOLSPARES files have been created on the volume and an unusable track is encountered (write error), the printer message is CYL nnn TR nnn UNUSABLE. An unusable track (a very unlikely possibility) has a bad surface and not even one bad-track track-linking record can be read from it.

Job Control File Halts

A B-address register value from 5000 through 5077 indicates an error condition related to statements of the job control file (see Table 2-6). The operator decides what action is required and enters the appropriate response character into the response location in the A-address register. The supplementary list is not produced.

Table 2-6. Job Control File Halts

B-Address Register Contents	Condition	Operator Action
5001	Invalid command field.	G = attempt to perform the operation again. The operator corrects the erroneous job control statement; refeeds the cards, starting with the Volume statement; enters a G and presses RUN. The program then reprocesses the job control file. E = exit to the Supervisor.
5002	Invalid positional parameter.	
5003	Invalid keyword.	
5004	Required parameter missing.	
5005	Invalid keyword parameter value.	
5010	Invalid combination or sequence of parameters.	
NOTE: G = 27_8, E = 25_8.		

Halts Specific to Volume Preparation C

A B-address register value from 6500 through 6517 indicates an error condition specific to Volume Preparation C.[1] See Table 2-7. The A-address register contains the address of the response location. The supplementary list is also present (see Table 2-4).

Table 2-7. Halts Specific to Volume Preparation C

B-Address Register Contents	Condition	Operator Action[1]
6500	The volume to be prepared is the volume on which the system resides.	G = refeed the job control file from the Volume statement, after changing the address of the volume to be prepared; enter G and press RUN. A = accept the situation and perform volume preparation on this volume. E = exit to the Supervisor.
6501	This halt occurs at the end of Volume Preparation C only when a 6500 halt has occurred earlier in the run and the A response has been given. A system residence volume must now be assigned to drive number 0 so that control can be transferred to the Supervisor.	If no legitimate system residence volume now exists, the run can be considered complete at this point and no response is necessary. G = reassign a system residence volume to drive number 0, enter G and press RUN. Volume Preparation C will return control to the Supervisor.

[1]Appendix B contains a description of halts in this category that apply to bad tracks on the volume being prepared.

Table 2-7 (cont). Halts Specific to Volume Preparation C

B-Address Register Contents	Condition	Operator Action[1]
6505	The volume-serial-number check failed.	G = reread the volume label. E = exit to the Supervisor.

[1]The operator enters the appropriate response character and presses RUN ($G = 27_8$, $A = 21_8$, $E = 25_8$).

CONSOLE MESSAGES

The console can be used for operator control when the Supervisor (in the Mod 1 (MSR) Operating System) or Floating Tape Loader-Monitor (in the Mod 1 (TR) Operating System) has been appropriately specialized and at least 16,384 characters of main memory are available. If either of these two conditions is not met, however, the operator's control medium is the control panel.

When the console is used for operator control, an error condition does not cause the system to halt. Instead, a diagnostic message is typed out on the console. The operator takes the following steps.

1. He reads the typeout. (To repeat the message, he presses the space bar twice.) If necessary, he consults the manual for possible action.
2. He performs the desired corrective action.
3. He types the appropriate 1-character response (G, E, etc.).
4. If the typein is correct, he presses the space bar to continue. If it is incorrect, he types any nonspace character and returns to step 3.

The routine continues as directed by the response code. If an invalid response is made, the routine causes the diagnostic message to reappear.

The following paragraphs provide detailed information about the three kinds of console messages that can occur during Volume Preparation C.

Peripheral Device Messages

Console messages relating to peripheral device error conditions are listed in Table 2-8. The format of these messages is

pp d MESSAGE TEXT

where pp = Peripheral control unit (leftmost bit set to zero),
d = Device number, and
MESSAGE TEXT specifies the error condition.

A second line contains the supplementary list described in Table 2-4.

Table 2-8. Peripheral Device Messages

Typewriter Message	Specific Code (alphanumeric)	Condition	Operator Action[1]

pp d INOPERABLE	1
	2
		Device inoperable.
		Protection violation.
			G = retry.
			E = exit to the Supervisor.

pp d READ ERROR		Read error (cards).	Correct erroneous card, if possible; refeed cards, starting with that card; and press the space bar.

pp d WRITE ERROR	4
	'
		Format violation.
		Track overflow.
			G = retry.
			E = exit to the Supervisor.

	8	Write error.[2]	G = retry.
			S = skip the bad track and resume processing.
			E = exit to the Supervisor.

pp d POSITIONING ERROR	3	Positioning error.	G = retry.
			E = exit to the Supervisor.

	=	Positioning error.	E = exit to the Supervisor.

[1]The operator types the appropriate alphabetic response character and presses the space bar.

[2]If *BADTRACKS and *VOLSPARES files have not been created on the volume and a bad track is encountered during formatting (write error), the following message appears on the printer: CYL nnn TR nnn ERROR (nnn is a decimal value). If *BADTRACKS and *VOLSPARES files have been created on the volume and an unusable track is encountered (write error), the printer message is CYL nnn TR nnn UNUSABLE. An unusable track (a very unlikely possibility) has a bad surface and not even one bad-track track-linking record can be read from it.

Job Control File Messages

All messages involving errors in the job control file begin with the words JOB CONTROL FILE ERROR. Table 2-9 contains the complete texts of all job control file messages for Volume Preparation C. No supplementary list is produced.

Table 2-9. Job Control File Messages

Typewriter Message	Condition	Operator Action
JOB CONTROL FILE ERROR, COMMAND FIELD	Invalid command field.	G = attempt to perform the operation again. The operator corrects the erroneous job control statement; refeeds cards, starting with the Volume statement; types a G and presses the space bar. The program then reprocesses the job control file. E = exit to the Supervisor.
JOB CONTROL FILE ERROR, POSITIONAL PARAMETER	Invalid positional parameter.	
JOB CONTROL FILE ERROR, KEYWORD PARAMETER	Invalid keyword.	
JOB CONTROL FILE ERROR, MISSING PARAMETER	Required parameter missing.	
JOB CONTROL FILE ERROR, PARAMETER VALUE	Invalid keyword parameter value.	
JOB CONTROL FILE ERROR, PARAMETER COMBINATION	Invalid combination or sequence of parameters.	

Messages Specific to Volume Preparation C

Table 2-10 contains messages for error conditions specific to Volume Preparation C.[1] The messages in Table 2-10 are followed by a second line of typed information, which contains the supplementary list described in Table 2-4.

[1]Appendix B contains messages in this category that apply to bad tracks on the volume being prepared.

Table 2-10. Messages Specific to Volume Preparation C

Typewriter Message	Condition	Operator Action[1]
SYSTEM RESIDENCE VOLUME WILL BE FORMATTED	The volume to be prepared is the volume on which the system resides.	G = refeed the job control file from the Volume statement, after changing the address of the volume to be prepared; type G and press the space bar. A = accept the situation and perform volume preparation on this volume. E = exit to the Supervisor.
REASSIGN SYSTEM RESIDENCE VOLUME TO DEVICE 0	This message occurs at the end of Volume Preparation C only when the above message has occurred earlier in the run and the A response has been given. A system residence volume must now be assigned to drive number 0 so that control can be transferred to the Supervisor.	If no legitimate system residence volume now exists, the run can be considered complete at this point and no response is necessary. G = reassign a system residence volume to drive number 0, type a G and press the space bar. Volume Preparation C will return control to the Supervisor.
INCORRECT VOLUME	The volume-serial-number check failed.	G = reread the volume label. E = exit to the Supervisor.

[1]The operator types the appropriate alphabetic response character and presses the space bar.

SECTION III

BOOTSTRAP GENERATOR C

Bootstrap Generator C creates a bootstrap routine on a mass storage volume. The bootstrap routine is the program that brings the Supervisor into main memory from mass storage at the beginning of operations with the Mod 1 (MSR) Operating System.

Bootstrap Generator C has the capability to perform the following:

1. Create the bootstrap routine;
2. Recreate the bootstrap routine at any time after its initial creation; and
3. Specialize the bootstrap routine for the requirements of a specific installation, so that it is possible to bootstrap the Supervisor without any halts for parameter changes.

Bootstrap Generator C is normally run when the operating system is generated for a particular installation. However, it can be run whenever it is necessary to condition a mass storage volume so that the Supervisor may be bootstrapped from it. This conditioning may be necessary if the installation is to run the Supervisor from more than one volume or to change some parameters of the bootstrap routine. In either case, the mass storage volume must first have been prepared with the Volume Preparation C routine.

Bootstrap Generator C has been designed to operate under the control of any of the monitors listed below: *

1. The Supervisor, or
2. Any loader-monitor in the Series 200/Operating System - Mod 1 (Tape Resident).

The reader should be familiar with the operating procedures for the Supervisor, as described in the Supervisor manual (Order No. 616).

EQUIPMENT REQUIREMENTS FOR BOOTSTRAP GENERATOR C

Basic Equipment Requirements

The equipment requirements for Bootstrap Generator C are listed below:

A Series 200 central processor;

Advanced Programming Instructions;

12,288 characters of main memory; and

One disk device and associated control.

This combination can be any one of the following.

Device Type	Control Type
155	157C, 257C
258	257, 257-1, 260
259	257, 257-1, 260
273	257, 257-1, 260
259A	257A
259B	257B
261	260
262	260

One card reader is also required.

Additional Usable Equipment

The following additional equipment also can be used:

One Type 220-1, -3 Console (console I/O requires 4,096 additional characters of main memory);

Second I/O Sector (Feature 1115 for Model 2200); and

One or more additional disk devices as tabulated above.

FUNCTIONAL DESCRIPTION OF BOOTSTRAP GENERATOR C

Bootstrap Generator C accepts the following parameters to create a bootstrap routine tailored for a particular Supervisor:

1. Unique Supervisor identification;

   ;upervisor's relocation bank indicator;

   \ddress of mass storage control from which the bootstrap routine will be xecuted;

   ddress of mass storage control and drive number of the volume on which ootstrap Generator C is to write the bootstrap routine; and

   ..sk transfer rate required at bootstrap time.

These parameters are specified through job control statements which are processed by the Bootstrap Generator C to specialize the bootstrap routine and write it onto a mass storage volume. Each of these parameters is described in more detail in the following paragraphs.

Supervisor Name

The Supervisor can exist in more than one version in the residence file. Each version is identified by the last character of the Supervisor's name.

During bootstrap generation, the user specifies which version of the Supervisor will be used most often and, therefore, will be loaded automatically by the bootstrap routine. This version of the Supervisor is referred to as the prime Supervisor. The term "prime" does not imply, however, that only one version of the Supervisor can be bootstrapped; the bootstrap procedure has an option to alter the Supervisor name and thus load another version. Also, the Supervisor can load any other version of itself through an Execute statement.

Location of Supervisor in Main Memory

Bootstrap Generator C can create a bootstrap routine that will relocate the Supervisor to a specific bank of main memory. This value can also be altered at bootstrap time.

Control Address for Bootstrap Routine

Bootstrap Generator C can specialize the bootstrap routine for a specific peripheral address assignment for the mass storage control. This value can also be altered at bootstrap time.

Device Address for Writing Bootstrap Routine

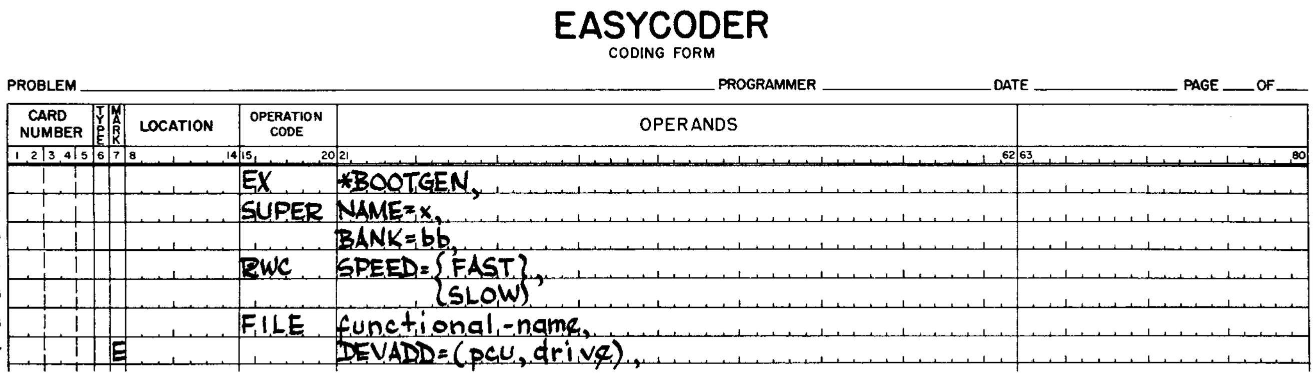
Bootstrap Generator C can write the bootstrap routine onto a volume having any mass storage control and device address. This parameter applies only at the time the bootstrap routine is generated. It allows Bootstrap Generator C to be loaded from one mass storage volume and to write the bootstrap routine onto another mass storage volume.

JOB CONTROL LANGUAGE FOR BOOTSTRAP GENERATOR C

Information to specialize the bootstrap routine is submitted through the job control file. The job control file must be assigned to a card reader.

Job control statements for Bootstrap Generator C (see Figure 3-1) are Execute, Supervisor, Read/Write Channel, and File. The Supervisor, Read/Write Channel, and File statements can be omitted; in which case certain values are assumed. (The Execute statement can be absent, or it can be replaced by a console call card if Bootstrap Generator C is not operated under the Mod 1 (MSR) Operating System. Refer to page 3-6 for operating procedures in a non-mass storage environment.)

To terminate the reading of job control statements, the character E must appear in the mark field of the last line of the last job control statement. The E must not appear in the Execute statement; therefore, if no other job control statements are submitted, a line containing E must still appear.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					EX	*BOOTGEN,	
2					SUPER	NAME=x,	
3						BANK=bb,	
4					RWC	SPEED={FAST},	
5						{SLOW}	
6					FILE	functional-name,	
7			E			DEVADD=(pcu,drive),	

Figure 3-1. Job Control Statements for Bootstrap Generator C

Execute Statement

The Execute statement directs the Supervisor to load Bootstrap Generator C. It must be the first statement in the job control file if Bootstrap Generator C is run under the control of the Supervisor. *BOOTGEN is the name of the Bootstrap Generator C program. A halt name may also appear (as described in the Supervisor manual).

Supervisor Statement

The Supervisor statement specializes the bootstrap routine with the name of the prime Supervisor and the relocation bank indicator. If the Supervisor statement is not supplied, Bootstrap Generator C assumes that:

1. The Supervisor name is SUPER1; and
2. The relocation bank indicator is 02 (specifying a memory size of 12K characters).

SUPERVISOR NAME PARAMETER

The Supervisor name parameter (NAME) specifies the last character of the name of the prime Supervisor. The format of this parameter is shown below.

```
NAME=x,
```

x - The last character of the name of the prime Supervisor. This character must be chosen from the letters (A-Z) or the digits (0-9). The entire name of the prime Supervisor is thus "SUPERx."

The default assumption is 1.

RELOCATION BANK INDICATOR PARAMETER

The relocation bank indicator parameter (BANK) specifies the bank in which the Supervisor is to reside. The format of this parameter is as follows.

```
BANK=bb,
```

bb — The relocation bank indicator, expressed as two octal characters. This parameter is chosen according to Table 3-1.

The default assumption is 02.

Table 3-1. Relocation Bank Indicator Parameter Values

Memory Size	Highest Location Used by the Supervisor (octal)	Relocation Indicator
12K	02 77 77	02
16K	03 77 77	03
20K	04 77 77	04
24K	05 77 77	05
28K	06 77 77	06
32K	07 77 77	07
40K	11 77 77	11
49K	13 77 77	13
57K	15 77 77	15
65K	17 77 77	17
81K	23 77 77	23
98K	27 77 77	27
114K	33 77 77	33
131K	37 77 77	37
163K	47 77 77	47
196K	57 77 77	57
229K	67 77 77	67
262K	77 77 77	77

RWC Statement

FAST gives an RWC variant of 56 or 76, depending on the sector bits specified in the FILE BOOT statement.

SLOW gives an RWC variant of 53 or 73.

The default value is FAST.

File Statements

File statements specify nonstandard control and device addresses for the two mass storage devices referred to by Bootstrap Generator C. These are:

1. Bootstrap Generator C device — the device onto which Bootstrap Generator C writes the bootstrap routine; and
2. Bootstrap device — the device to be used later for the bootstrap operation.

If no File statements are supplied, these control and device addresses are assumed to be:

1. Bootstrap Generator C device — control 04, drive 0; and
2. Bootstrap device — control 04 (drive number is not relevant).

FUNCTIONAL-NAME PARAMETER

The functional-name parameter identifies the function to which the File statement applies. This parameter has the following format.

$$\boxed{\left\{\begin{matrix}\text{GEN}\\\text{BOOT}\end{matrix}\right\},}$$

GEN - Bootstrap Generator C device — the device onto which Bootstrap Generator C writes the bootstrap routine.

BOOT- Bootstrap device — the device to be used for subsequent bootstrap operation. For this device, the drive number is not relevant, since the bootstrap operation is always from drive 0.

DEVICE-ADDRESS PARAMETER

The device-address parameter (DEVADD) specifies the peripheral control unit (pcu) address and drive number to be assigned for this File statement. This parameter is required if the File statement is present. This parameter has the following format.

```
DEVADD = (pcu, drive),
```

pcu - Address of peripheral control unit, written as two octal characters. The value of the leftmost bit is insignificant.

drive - Drive number, written as one octal character. This parameter is not significant for the bootstrap device (functional-name is BOOT) and can be omitted.

OPERATING PROCEDURES FOR BOOTSTRAP GENERATOR C

Preparing the Volume for Bootstrap Generation

In order to generate the bootstrap routine, the mass storage volume must have been prepared by the Volume Preparation C routine to contain the following information:

1. Cylinder 0, track 0 formatted with 250-character records;
2. Volume label; and
3. Volume directory.

Loading Bootstrap Generator C

The procedures for loading Bootstrap Generator C depend on the monitor and the operating system being used. Bootstrap Generator C can be loaded using any one of the following monitors.

1. Mass storage Supervisor.
2. One of the following loaders in the Series 200/Operating System - Mod 1 (Tape Resident).
   a. Tape Loader-Monitor C; or
   b. Floating Tape Loader-Monitor C.

LOADING WITH THE SUPERVISOR

The statements in the job control file are exactly as described on page 3-3 under the heading "Job Control Language for Bootstrap Generator C," i.e., an Execute statement followed by one Supervisor statement and up to two File statements. The Supervisor and File statements can appear in any order or can be absent. In any case, the last line of the last job control statement must contain an E in the mark field.

The following procedures are used to load with the Supervisor:

1. Place the job control statement in the card reader;
2. Obtain the Supervisor "ready state," i.e., the Supervisor is ready to read an Execute statement; and
3. Press RUN (if necessary).

LOADING WITH A MOD 1 TAPE LOADER-MONITOR

When loading with a Mod 1 tape loader-monitor, the Execute statement is replaced by a console call card. The format of the console call card is shown in Table 3-2. The remainder of the job control file is the same as that for the Mod 1 (MSR) Operating System.

Table 3-2. Format of Console Call Card

Characters	Contents	Meaning
1 - 8	*BOOTGEN	Program and segment names of Bootstrap Generator C.
9	d	Tape drive number from which Bootstrap Generator C is to be loaded.
10 - 17	ppppppss	The Supervisor halts after the named segment is loaded. If left blank, the monitor does not halt.
18	*	Asterisk (required by the Mod 1 (TR) Operating System).

The following procedures are used to load with a Mod 1 tape loader-monitor:

1. Place the job control file in the card reader;
2. Obtain halt 17002 (monitor ready to read a console call card);
3. If slow disk transfer rate only is required at generation time, display location 63 (77 octal) and replace with an item mark; and
4. Press RUN.

OPERATOR CONTROL AND MESSAGES

At the time of a condition requiring operator action, information is conveyed to the operator in the following manner:

1. Through coded values in the B-address register if the control panel is being used; or
2. Through typed messages if the console is being used.

The operator takes corrective action and:

1. Enters the response character through the control panel if the control panel is being used; or
2. Types the response character at the console keyboard if the console is being used.

Control Panel Halts

When a halt occurs, the B-address register contains a coded value that indicates the reason for the halt. Normally, this is sufficient for the operator to take appropriate action.

In some cases, the A-address register value supplies supplementary information to the operator at the time of the halt. In such cases, the A-address value is the address of a list of information. The contents of this list depend on the specific condition and are defined in the following paragraphs.

In the tables below only the rightmost four octal digits (12 bits) of the B-address value are shown. The remaining bits are zeros.

PERIPHERAL DEVICE CONDITIONS

A B-address register value in the range 0000 to 3777 (inclusive) indicates a condition related to a peripheral device operation. The B-address register value has the general form ppxd:

pp	= Peripheral control address assignment with the leftmost bit set to zero;
x	= Code indicating the type of peripheral condition; and
d	= Device number (if applicable).

If the device is mass storage, the A-address register value is the address of a list of information in the following order.

1.	Response location	- One character.
2.	Specific code	- One character, indicating supplementary information about the condition.
3.	File name	- Ten characters, giving the name of the file being processed.

The operator should determine which peripheral device is referred to by the "pp" and "d" values of the B-address register, and then he should consult the following table.

Table 3-3. Halts Caused by Peripheral Device Conditions

B-Address Register Value	Specific Code	Condition	Operator Action
pp0d		Device inoperable (mass storage only)	See specific action below.
	01	Device inoperable	Cycle up device if necessary. Enter G and press RUN to continue.
	02	Protection violation	Verify that protection switches on control are set properly. Enter G and press RUN to continue.
pp1d		Uncorrectable read error	See specific action below.

If pp = card reader:		d = 1 cycle check
		d = 2 validity check
			Correct card if possible. Refeed cards starting with card in error. Press RUN to continue.

pp2d		Uncorrectable write error	See specific action below.
If pp = mass storage:	04 10		Enter G and press RUN to try again. If error persists, record specific code and terminate run.
pp4d	03 05	Positioning or addressing error (mass storage only)	Enter G and press RUN to try again. If error persists, record specific code and terminate run.
pp7d	11	Miscellaneous device error condition (mass storage only)	

JOB CONTROL FILE CONDITIONS

A B-address register value in the range 5000 to 5077 indicates a condition related to the job control file. The A-address register value is not used when the halt is caused by a job control file condition.

The following table contains the B-address register values, their meanings, and corresponding operator actions related to job control file conditions.

Table 3-4. Job Control File Condition Halts

B-Address Register Value	Meaning	Operator Action
5001	Command code not recognized	Correct the card in error and refeed starting with that card. Press RUN.
5003	Keyword not recognized	
5005	Illegal value for keyword parameter	

BOOTSTRAP GENERATOR C END-OF-RUN

A B-address register value of 6520 indicates that the bootstrap routine has been written on the mass storage device. This halt is necessary because Bootstrap Generator C may be run as a self-loading card deck, in which case a loader halt, signifying end-of-run, is not available. The following table gives the meaning and operator action for the 6520 halt.

Table 3-5. Bootstrap Generator C End-of-Run

B-Address Register Value	Meaning	Operator Action
6520	End-of-run	If operating under a resident monitor, press RUN to get to a 'ready state (or 17002 halt).

Console Messages

When a console message indicates an error or requests operator action, the operator performs the following steps:

1. He reads the typeout. (To repeat the message, he presses the space bar twice.) If necessary, he consults the manual for possible action.
2. He performs the desired corrective action.
3. He types the appropriate 1-character response (G, E, etc.).
4. If the typein is correct, he presses the space bar to continue. If it is incorrect, he types any other character and returns to step 3.

PERIPHERAL DEVICE CONDITIONS

When the first line of a message typed on the console is of the form

pp d message

then the condition causing the message is related to peripheral device operation.

pp	= Address assignment of peripheral control (with the leftmost bit set to zero);
d	= Device number (if applicable).

If the device is mass storage a second message line is typed. This line is a list of information, in the following order:

Specific code	- One character, indicating supplementary information about the condition and
File name	- Ten characters, giving the name of the file being processed.

The operator should determine which peripheral device is referred to by the "pp" and "d " portions of the message; then Table 3-6 should be consulted to determine appropriate action.

The operator must enter a response character (i.e., type in a character when the TYPE light goes on).

Table 3-6. Peripheral Device Condition Messages

Typewriter Message	Specific Code (alphanumeric)	Condition	Operator Action
pp d INOPERABLE		Device inoperable (mass storage only)	See specific action below.
	1	Device inoperable	Cycle up device, if necessary. Enter G to continue.
	2	Protection violation	Verify that protection switches on control are set properly. Enter G to continue.
pp d READ ERROR		Uncorrectable read error	See specific action below.
		d = 1 cycle check	Correct card, if possible. Refeed cards, starting with card in error.
		d = 2 validity check	Enter G to continue.
pp d WRITE ERROR		Uncorrectable write error	See specific action below.
	4 8		Enter G to try again. If error persists, record specific code and terminate run.

Table 3-6 (cont). Peripheral Device Condition Messages

Typewriter Message	Specific Code (alphanumeric)	Condition	Operator Action
pp d POSITIONING ERROR	3 5	Positioning or addressing error (mass storage only)	Enter G to try again. If error persists, record specific code and terminate run.
pp d MISCELLANEOUS	9	Miscellaneous device error condition (mass storage only)	

JOB CONTROL FILE CONDITIONS

When the line typed out is of the form,

JOB CONTROL FILE ERROR, message

then the condition demanding a response is related to the job control file. No second line is typed.

The diagnostic messages are listed in Table 3-7.

Table 3-7. Messages for Job Control File Conditions

Typewriter Message	Meaning	Operator Action
JOB CONTROL FILE ERROR, COMMAND FIELD	Command code not recognized.	Correct the card in error and refeed starting with that card.
JOB CONTROL FILE ERROR, KEYWORD PARAMETER	Keyword not recognized.	
JOB CONTROL FILE ERROR, PARAMETER VALUE	Illegal value for keyword parameter.	Enter G to continue.

BOOTSTRAP GENERATOR C END-OF-RUN

Table 3-8 contains the typewriter message, which indicates that the bootstrap routine has been written on the mass storage device. Operator action is necessary because Bootstrap Generator C may be run as a self-loading card deck, in which case there is no resident loader to return to.

Table 3-8. Bootstrap Generator C End-of-Run Message

Typewriter Message	Meaning	Operator Action
END JOB	End-of-run	If operating under a resident monitor, enter G to get to a ready state.

SECTION IV

MASS STORAGE SORT C

INTRODUCTION TO MASS STORAGE SORT C

Mass Storage Sort C (herein after referred to as the sort) is a utility routine of the Series 200/Operating System - Mod 1 (Mass Storage Resident). It is highly flexible, enabling it to meet a wide variety of user requirements. It may be operated as a key sort, an extract sort, or an item sort. Accompanying the sort are two routines which facilitate the processing of the sorted output. These two routines are: the Fetch macro routine (similar to an input/output macro routine) and the sequential output file program. The relationship between the sorting process and these two output processes is discussed below.

A key sort takes the sort-key fields from the input item (source item) and attaches the mass storage address of the item to the key. It then processes only this condensed data, which saves considerable time if the key is a small percentage of the total item and the file is fairly large. The sort output (sort-item file) is an ordered collection (via chaining) of the keys with the appended addresses of the associated items. The user may write a program incorporating the Fetch macro statement and process the output of the sort. The Fetch routine retrieves certain elements of the sorted output sequence as follows: first, the sort item containing the key and the source-item address, and second, the associated mass storage resident source item. Similarly, the sequential output file program can be used to create a sequential file (as defined in the Data Management Subsystem manual) of the source items in the sorted output sequence.

An extract sort is similar to a key sort except that selected fields are extracted from the source item and carried through the sort process with the key. If the application is concerned only with selected fields of the source item, considerable time can be saved during the extract option. For example, if a file is being sorted in preparation for producing a report and if the key, plus the fields required for the report, are a small part of the full source item, an extract sort can be highly efficient. The user may access the key and the extracted fields in the sort output sequence either with own-coding during the final merge portion of the sort or with the Fetch routine in a program of his own. In either case, there is no need to reaccess the source items. The sequential output file program can be used to create a sequential file of sort items. (Each item in the file is made up of fields of the corresponding source item.)

An item sort is essentially an extract sort where the entire source item is extracted. In the sort, the item sort option produces a sort item (prior to the Fetch operation) which is in the same form as the source item. (In the case of the key sort or the extract sort, the sort item is in the rearranged form shown in Figure 4-12.)

Figure 4-1 illustrates the relationship between the sorting process and the output processes. Note that the sort-item file is always input to the output processes. The source-item file is input only if access to the source items is required.

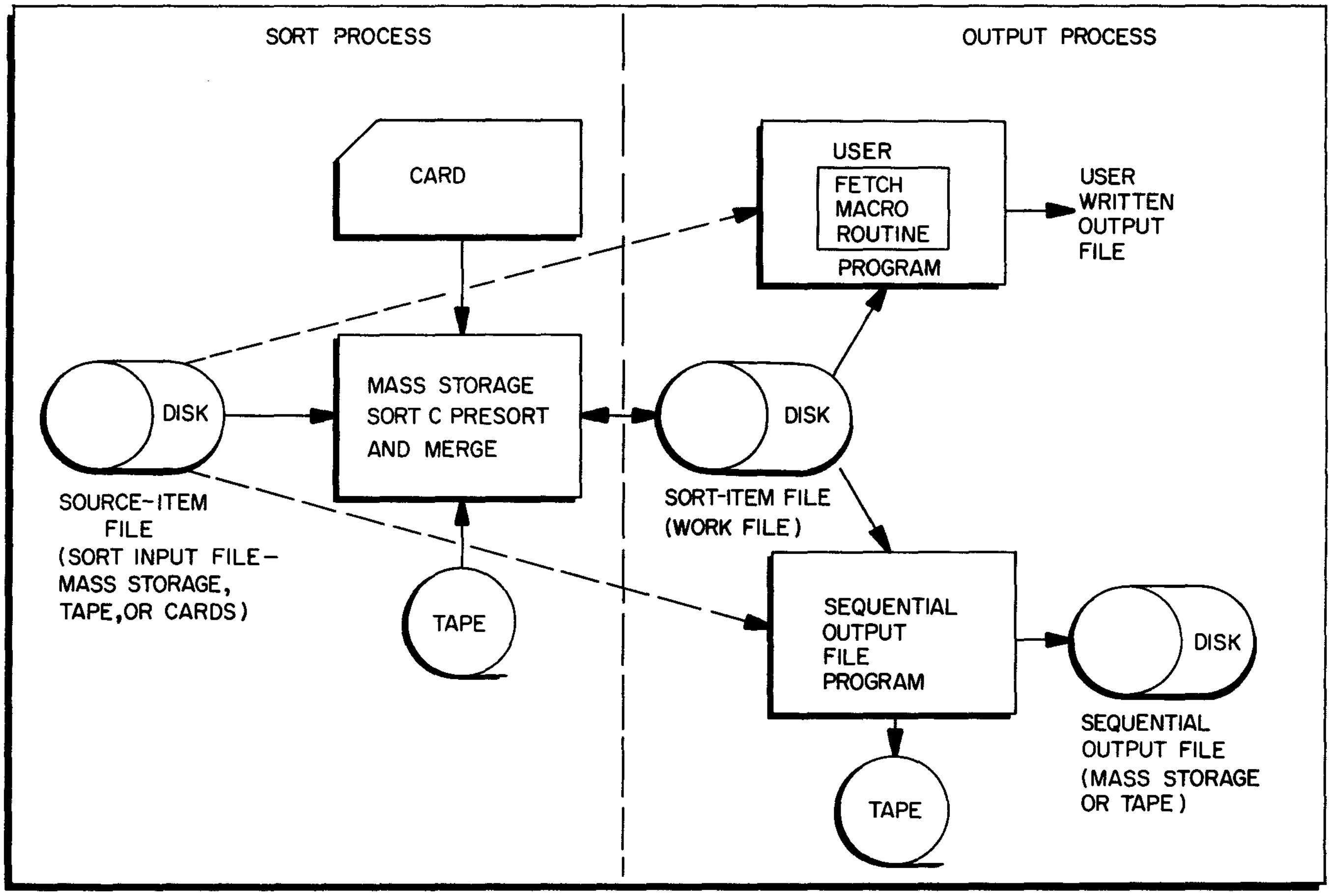

SORT PROCESS
OUTPUT PROCESS
CARD
USER
FETCH MACRO ROUTINE
PROGRAM
USER WRITTEN OUTPUT FILE
DISK
MASS STORAGE SORT C PRESORT AND MERGE
DISK
SOURCE-ITEM FILE (SORT INPUT FILE-MASS STORAGE, TAPE,OR CARDS)
SORT-ITEM FILE (WORK FILE)
TAPE
SEQUENTIAL OUTPUT FILE PROGRAM
DISK
SEQUENTIAL OUTPUT FILE (MASS STORAGE OR TAPE)
TAPE

Figure 4-1. Relationship Between Sorting Process and Output Processes

EQUIPMENT REQUIREMENTS FOR MASS STORAGE SORT C

Basic Equipment Requirements

The equipment requirements for Mass Storage Sort C are listed below:

A Series 200 central processor;

Advanced Programming Instructions;

12, 288 characters of main memory; and

One disk device and associated control.

This combination may be any one of the following:

Device Type	Control Type
155	157C, 257C
258	257, 257-1, 260
259	257, 257-1, 260
273	257, 257-1, 260
259A	257A
259B	257B
261	260
262	260

One card reader (unless sort parameters are resident in memory) is also required.

Additional Usable Equipment

The following additional equipment also may be used:

One printer;

One or more 1/2-inch Type 204B Magnetic Tape Units;

One Type 220-1 or -3 Console (console I/O requires 4,096 additional characters of main memory);

Up to a total of 32,768 characters of main memory;

Second I/O Sector (Feature 1115 for Model 2200); and

One or more additional disk devices as tabulated above.

SUMMARY OF MASS STORAGE SORT C CAPABILITIES

The following list summarizes capabilities of the sort.

1. Sorts fixed-length items.
2. Sorts according to sort-key fields up to a maximum of ten in ascending, descending, or mixed sequence.
3. Allows up to 11 extract fields or residue of an item to be an element of the sort item.
4. Allows translation which permits the sorted order to be based upon the commercial collating sequence.
5. Permits the selection of only those items for the sort that have a field that bears a relationship to a specified value.
6. Permits deletion from the sort of those items that have a field that bears a relationship to a specified value.
7. Allows user's own-coding on an input item-by-item basis.
8. Allows user's own-coding on an output sort item-by-item basis.
9. Accepts input from a mass storage device, magnetic tape, cards, or through own-coding only.
10. Can process sequential, direct access, and indexed-sequential mass storage files.
11. Provides output in a work-file area on the mass storage device.

12. Allows the user to restrict the sort-item block size.
13. If requested, it ensures that the work area defined by the user is sufficient for the successful completion of the sort.
14. Permits the user to receive the results of sort-item block and work-area calculations prior to and apart from sort execution.
15. Preserves the original input file.
16. If requested, it creates a sequential file on mass storage or a file on 1/2-inch magnetic tape.

FUNCTIONAL DESCRIPTION OF MASS STORAGE SORT C

Definition of Terms

The following terms are used frequently in this section and are hereby defined for the reader.

1. Sort keys — those fields of an item that determine the order of the sort output and are part of the sort output.
2. Extract fields — those fields of an item that may be selected to be part of the sort output. Extract and sort-key fields must be distinct.
3. Sort item — the internal item of the sort that has been derived from a source item. It will contain sort-key fields and may contain extract fields and a source-item address.
4. Residue — that data of an item that is not contained in a sort-key field.
5. Fetch — a macro routine that simplifies the processing of the sort item file as well as the input file from which the sort file was generated unless that file was on card or tape. The Fetch macro routine is described in detail on page 4-6.

Function of Program

The sort processes a file stored on either a mass storage device, a 1/2-inch magnetic tape, or punched cards. It also processes items entered through own-coding only. A mass storage file that is to be input to the sort must conform to the Mod 1 (MSR) data management conventions for either a sequential, direct access, or an indexed sequential file. Input from cards and 1/2-inch magnetic tape are discussed on page 4-8.

From each item that is a valid entry to the sort, a sort item is developed which contains the sort-key fields as a unit. The address of the mass storage resident source item may be appended to the sort-key unit, and the extract or residue may be prefixed to that same unit.

The sort produces one string or file of logically sequenced sort items in a work area. Ordering in the logical sequence is a function of the Honeywell collating sequence (binary 000000 to 111111) as well as the sort-key fields. A request for translation causes the sorted order to be based on the commercial collating sequence. A user requiring some other collating sequence is able to achieve it through own-coding. At the completion of the sort process, the sort items may then be made available to the user through a specialized Fetch macro routine which may retrieve an associated mass storage resident source item.

If the user desires, the output from the sort may be a sequential file, organized according to the data management conventions, or it may be a file on 1/2-inch magnetic tape. Refer to "Sequential Output File Program Input/Output Files," in this section.

Function of Segments

The sort may be considered as consisting of two logical segments: presort and merge.

PRESORT SEGMENT

The presort segment develops a sort item from each item that is acceptable to the particular application. These sort items are arranged into ordered strings whose length is determined by the available memory and the requirement that an efficient covering is made of the available cylinder area.

Own-coding can be entered during the presort, thus permitting the inspection, modification, deletion, and addition of items.

MERGE SEGMENT

The merge segment is subdivided into two phases, "1-cylinder," and "multicylinder."

The 1-cylinder phase merges the strings that exist on one cylinder until they are reduced to a single string of sort items. Memory available to the 1-cylinder phase is a factor in determining the "way" of the merge.

The multicylinder phase merges two cylinders until a final string is produced. A sliding buffer technique is employed to ensure the best activity on a cylinder once it has been accessed. This phase utilizes the available memory by creating as many buffers as it can. When the final string is being created, any required reorganization of the sort item is accomplished. If merge own-coding is requested, it becomes active during this final phase and permits access to the sort item after it has been reorganized.

If the type of output requested from the sort is in the form of either a sequential file or a file on 1/2-inch magnetic tape, the multicylinder phase calls the sequential output file program (see page 4-9).

FUNCTIONAL DESCRIPTIONS OF OUTPUT PROCESSES

Functional Description of Fetch Macro Routine

The Fetch macro routine simplifies the processing of the Mass Storage Sort C sort-item file as well as the mass storage resident source-item file from which the sort-item file was generated. It makes available the sort item and its associated source item on an item-by-item basis. Hence, the items of the mass storage resident source-item file can be processed in the logical sequence promoted by the sort application. The user may exercise options that restrict his access to either the sort items or to the mass storage resident source items.

SPECIALIZATION OF FETCH MACRO ROUTINE

Partial specialization of the Fetch macro routine is achieved principally by assigning values to the parameters of the Fetch macro statement. However, certain parameters required for full specialization can be ascertained only after the sort has produced its sort-item file. The values for these parameters are written by the sort on a mass storage device. When the program containing the partially specialized version of Fetch is executed, the block containing these values is accessed by Fetch and the final specialization is achieved. A user may modify this final version by requesting an exit before any file processing begins.

A detailed description of the Fetch macro routine is dicussed under "Language Elements of Fetch Macro Routine," in this section.

Functional Description of Sequential Output File Program

The sequential output file program processes the Mass Storage Sort C sort-item file and, when necessary, the mass storage resident file that was input to the sort (source file). It creates either a sequential file on mass storage or a file on 1/2-inch magnetic tape containing the items in the sequence produced by the particular sort application.

A detailed description of the sequential output file program is discussed under "Job Control Language for Sequential Output File Program," in this section.

When the request for a sequential output file or a tape file is made by the inclusion of an output file statement in the sort job control file, the sort creates an additional parameter record in the sort work file. The contents of this parameter record are made available to the sequential output file program when it is called by the sort.

When the sequential output file program is called through a separate Execute statement, the parameter information must be submitted to the sequential output file program through a separate set of job control statements.

The sequential output file program consists of two types of processing.

SORT–ITEM FILE TO SEQUENTIAL FILE

This process is called if in the preceding sort application no source-item address was appended to the sort item, or an item sort was executed. The assumption is that the sequential file or tape file to be created will have an item that corresponds to the sort item as it exists in the sort-item file created by the sort. (When an item sort is requested, the sort item is the complete source item.)

When item own-coding is specified, the sort item is made available to the user before it is transferred to item storage. The item may be deleted or modified. Alternatively, an item having the same characteristics as the sort item may be added.

SOURCE–ITEM FILE TO SEQUENTIAL FILE

This process is called if the sort item has a mass storage source-item address appended to it. It is assumed that the items to be placed in the output file are to be obtained from a mass storage resident source file (original input file).

When item own-coding is specified, the sort item is made available to the user. Note that the sort item is determined by the values given to the key and extract parameters of the sort. The user may now delete an item, based upon the contents of the sort item, or an item having the same characteristics as the source items may be added for inclusion in the output file.

FUNCTIONAL DESCRIPTION OF INPUT/OUTPUT FILES

Normally, a sort is run to produce a sorted sequence of keys, extracts, or items. The sort-item file is then processed either by a program containing a specialized Fetch macro routine and user coding or by the sequential output file program. The latter may be called through a separate Execute statement, or it may be called directly by the sort through the use of an output file statement in the job control file. In either case, the sort-item file is the input to the Fetch macro routine or to the sequential output file program.

Sort Input/Output Files

The input file (source file) to the sort may reside either on a mass storage device, on 1/2-inch magnetic tape, or on punched cards. In addition, all items to be sorted may be entered through own-coding.

A mass-storage-resident source file may be any file organization (except partitioned sequential) that conforms to the Mod 1 (MSR) data management conventions.

A file on 1/2-inch magnetic tape that is to be processed by the sort must contain data records consisting only of fixed-length items. Also, the file must conform to one of the following tape file formats:

1. Bannerless, standard labels in odd or even parity;
2. Bannerless, no labels in odd or even parity;
3. Bannered, standard labels in odd or even parity; and
4. Bannered, no labels in odd or even parity.

In the case of standard labels, all headers are treated as if they are regular and not extended; therefore, information in character positions 41-80 in the header label is disregarded. Tape marks may occur following the header and/or preceding a standard trailer. The presence of these tape marks on the input tape is recorded if a request is made for tape output through the sort job control file. Tape input files may be multireel, and data records may be blocked or unblocked.

Card input to the sort may be read in standard or in special mode. The first card in the input deck should contain the first item, or part of the first item. When the item exceeds 80 characters, it must exist on consecutive cards. The first character of an item must start in column 1 of the card. For example, consider an item 120 characters in length as shown in Figure 4-2.

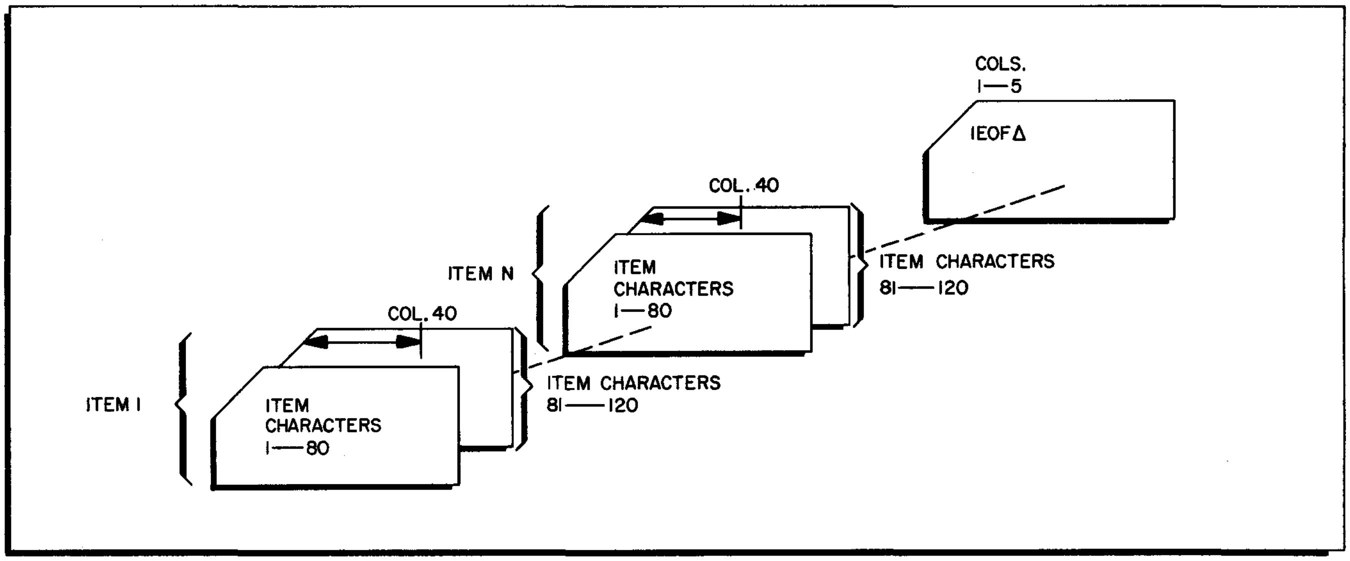


Figure 4-2. Card Input File

Note that Figure 4-2 shows that the last item must be followed by a card containing 1EOFΔ in columns 1-5. All input may be through own-coding as described under "Own-Coding-Only Source of Input to the Sort," in this section.

The output from the sort (sort-item file) is an aggregate of chained blocks containing sort items in the sorted sequence. This sort-item file uses the area specified to the sort as the work files. However, because the structure of this file does not conform to the mass storage data management conventions, it should be processed by either the Fetch macro routine or the sequential output file program. Figure 4-1 illustrates the relationship between the sorting process and the output processes. The format of the sort items for each type of sort (key, extract, or item) is shown in Figure 4-12.

Fetch Macro Source-Item/Sort-Item Files

The sort-item file is always an input file to the program containing the Fetch macro routine (see Figure 4-1). This file is used to access the data of the sort application in the sorted sequence. If the application is an extract sort or an item sort, then only this file is needed as input to the Fetch macro routine.

If the source file is mass storage resident and the application requires access to the original input file (as is necessary in a pure key sort), the source item is also an input to the Fetch macro routine. Items in this file are accessed "directly" in the sorted sequence, as directed by the items in the sort-item file.

If the user desires, he may create a file containing sort items by using the program containing the Fetch macro routine. This process is entirely under user control.

Sequential Output File Program Input/Output Files

The sort-item file is always one input file to the sequential output file program, as it is for the Fetch macro routine. If no source-item address is appended to the sort item, the sequential output file program cannot access the source file. The sort item becomes the final output item.

If a source-item address is appended to the sort item, the source file resides on a mass storage device and is a second input file to the sequential output file program. The source item becomes the final output item.

The sort-ordered output file from the sequential output file program may be either a sequential file on a mass storage device or a file on 1/2-inch magnetic tape.

When the output file request is made at sort execution time, the creation of the output file may be regarded as a process executed by a logical segment of the sort. If this is the case, and the input to the sort is 1/2-inch magnetic tape, the format of the output tape is the same as that of the input tape, with the exception that the size of the data record may be changed. Tape marks present on the input tape appear on the output tape. Input tape labels are updated and used as output labels. If the input to the sort comes from a medium other than magnetic tape and tape output is requested, the format of the output tape is bannered, standard labels in odd parity. A dummy header and trailer are created.

When tape output is requested through the job control file for the sequential output file program, the following file formats may be created:

1. Bannerless, standard labels in odd or even parity;
2. Bannerless, no labels in odd or even parity;
3. Bannered, standard labels in odd or even parity; and
4. Bannered, no labels in odd or even parity.

If the tape input to the preceding sort has standard labels and standard labels are requested on the output tape, the input labels are updated and used as the output labels. If the input to the sort comes from some medium o+her than tape or from a tape with no labels, and tape output with standard labels is requested, dummy labels are created by the sequential output file program.

Table 4-1 summarizes the various tape output file formats created by the sequential output file program.

Table 4-1. Tape Output File Formats Created by Sequential Output File Program

Output / Input	Sequential Output File Program - Called for Execution by the Sort	Sequential Output File Program - Called by an Execute Card (LABELS=YES,)	Sequential Output File Program - Called by an Execute Card (LABELS=NO,)
Tape Input (LABELS= YES,)	1. Input labels updated and used as output labels. 2. Format of output tape same as input tape with regard to TM, BAN, PAD, PAR. 3. Size of output data record must be specified in job control file. 4. File name is changed only if specified in sort job control file. 5. Input retention cycle preserved.	1. Input labels updated and used as output labels. 2. Format of output tape (TM, BAN, PAD, PAR) specified through job control file. 3. File name and record independent of input file (specified through job control file). 4. Input retention cycle preserved.	1. No labels. 2. Format of output tape [(TM), BAN, PAD, PAR] specified through job control file. 3. TM automatically placed after data. 4. Output record size specified through job control file.
Tape Input (LABELS=NO,)	1. No labels. 2. Format of output tape same as input tape with regard to TM, BAN, PAD, PAR. TM automatically placed after data. 3. Size of output data record must be specified in sort job control file.	1. Create dummy header and trailer. 2. Format of output tape (TM, BAN, PAD, PAR) specified through job control file. 3. File name and record size independent of input file (specified through job control file). 4. Retention cycle 0.	Same as above.
Other Input Disk Cards Own-Coding Only	1. Create dummy header and trailer. 2. Format of output tape is bannered, odd parity and padding of 77_8. The banner is 56_8. No tape marks appear.	Same as above.	Same as above.

JOB CONTROL LANGUAGE FOR MASS STORAGE SORT C

Requesting Sort Operation

The specification of the parameters to the sort may be made through statements in the job control file, or the parameters may be resident in main memory when the sort is called. (The job control file must be in the card reader. For a job control file in a card reader, the term "line" denotes one card.)

The two possible Execute statements are described under "Loading and Executing," in this section.

Job Control Statements and Parameters

A variable number of job control statements is required to define the sort parameters, depending on the demands of a particular application.

OPTIONAL PARAMETERS

Optional parameters have an assumed or "default" value. When such a parameter is omitted in the job control statements, the default value is used.

PARAMETER SET

The collection of statements required for a sort operation is called the parameter set. An E in the mark field of a line indicates that this is the last line of the set. The line containing the E may be the same line as the last line containing a parameter, or it may be a separate line containing no parameters.

ORDER OF STATEMENTS

Statements may be in any order in the job control file. The first line of a statement must have a nonblank operation code field.

All device address parameters (for multivolume files) should be written on a single line to reduce difficulties if card decks are disarranged.

NUMERIC VALUES

Numeric values in parameters are always decimal (unless otherwise noted), and leading zeros may be omitted.

Sort Statement

A Sort statement is required in the job control file; the format of this statement is shown in Figure 4-3.

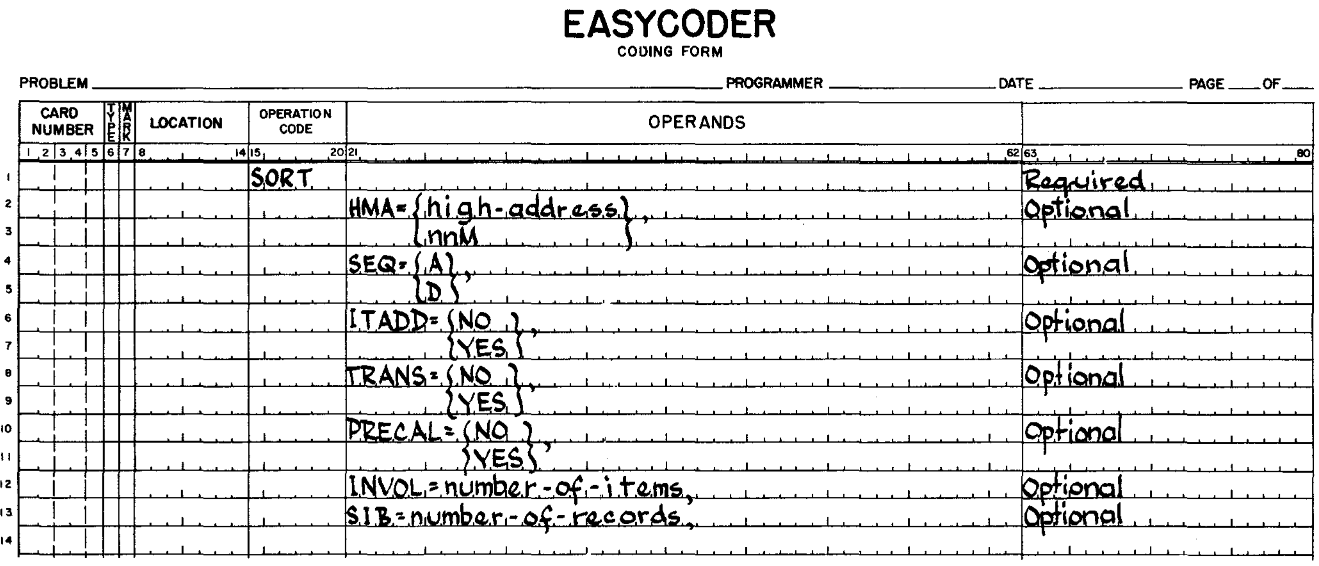
HIGH-MEMORY-ADDRESS PARAMETER

The high-memory-address parameter (HMA) defines the highest memory address available to the sort. However, if an own-code program is resident during a phase of the sort and the base address of that program is lower than that specified by the HMA parameter, then the base address of that program less one is the highest memory address available to that phase. This parameter takes the following form.

$$\boxed{\text{HMA} = \left\{ \begin{matrix} \text{high address} \\ \text{nnM} \end{matrix} \right\},}$$

<u>high-address</u>	- The highest address available, written as a decimal number (up to six characters in length).
<u>nnM</u>	- The number of memory modules (M) available, written as a 2-character decimal number. The highest address available is computed by the sort and is one location less than nn times 4, 096. M is a parameter constant.

The default assumption is that the HMA value is to be taken from the Supervisor communication area field that specifies the highest address available to the object program memory. If HMA is specified but has a higher address than the address in the Supervisor, the Supervisor value is used.


EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1					SORT		Required
2						HMA={high-address},	Optional
3						{nnM}	
4						SEQ={A},	Optional
5						{D}	
6						ITADD={NO},	Optional
7						{YES}	
8						TRANS={NO},	Optional
9						{YES}	
10						PRECAL={NO},	Optional
11						{YES}	
12						INVOL=number-of-items,	Optional
13						SIB=number-of-records,	Optional
14							

Figure 4-3. Sort Statement and Parameters

SEQUENCE PARAMETER

The sequence parameter (SEQ) describes the primary sorting sequence. The form of the parameter is as follows.

$$\boxed{\text{SEQ} = \left\{ \begin{matrix} \text{A} \\ \text{D} \end{matrix} \right\},}$$

<u>A</u> - Ascending sequence.

<u>D</u> - Descending sequence.

The default assumption is ascending sequence.

ITEM-ADDRESS PARAMETER

The item-address parameter (ITADD) indicates whether or not the mass storage address of the item is to be appended to the sort item. The form of this parameter is as follows.

$$\boxed{\text{ITADD} = \left\{\begin{matrix}\text{NO}\\ \text{YES}\end{matrix}\right\},}$$

<u>NO</u> - The item address is not to be appended.

<u>YES</u> - The item address is to be appended.

The use of the item-address parameter when input is from cards or tape is meaningless; the default assumption for these two input media is NO. If the input is from mass storage or through own-coding, the default assumption is YES. Use YES if the application requires accessing of the input file after completion of the sort (refer to "Detailed Description of Fetch Macro Routine" and to "Detailed Description of Sequential Output File Program in this section).

TRANSLATION PARAMETER

The translation parameter (TRANS) indicates the sort order; it is to be in either Honeywell collating sequence (binary 000000 to 111111) or commercial collating sequence.

$$\boxed{\text{TRANS} = \left\{\begin{matrix}\text{NO}\\ \text{YES}\end{matrix}\right\},}$$

<u>NO</u> - Translation is not required, and the Honeywell collating sequence is to be used.

<u>YES</u> - Translation is required, and the commercial collating sequence is to be used.

The default assumption is NO.

PRECALCULATION PARAMETER

The precalculation parameter (PRECAL) indicates whether the sort is to be executed after the calculation of the sort-item block size and, if requested, the determination that sufficient work area has been allocated.

$$\boxed{\text{PRECAL} = \left\{\begin{matrix}\text{NO}\\ \text{YES}\end{matrix}\right\},}$$

<u>NO</u> - The sort is to be executed.

<u>YES</u> - The sort program is to return to the Supervisor after completing the requested calculation.

The default assumption is that the sort is to be executed. When the parameter value is YES, the sort indicates the results of the calculations through the printer, if available, or, if the printer is not available, through the operator control file.

VOLUME OF INPUT PARAMETER

This parameter (INVOL) indicates to the sort that, prior to execution, a check is to be made to ensure that the work area defined by the specification of work files is sufficient for the successful execution of the sort.

```
INVOL = number-of-items,
```

number-of-items - The number of items to be processed by the sort (six decimal characters). Leading zeros not required.

The default assumption is that the size of the work area is not checked.

SORT-ITEM BLOCK RESTRICTION PARAMETER

The sort-item block restriction parameter (SIB) indicates that the user wishes to control the maximum size of the sort-item block.

```
SIB = number-of-records.
```

number-of-records - The maximum number of 250-character physical records that the sort may use to contain the sort item block (a 2-character decimal number).

The default assumption is that the sort can freely determine the size of its sort-item block.

File Statements

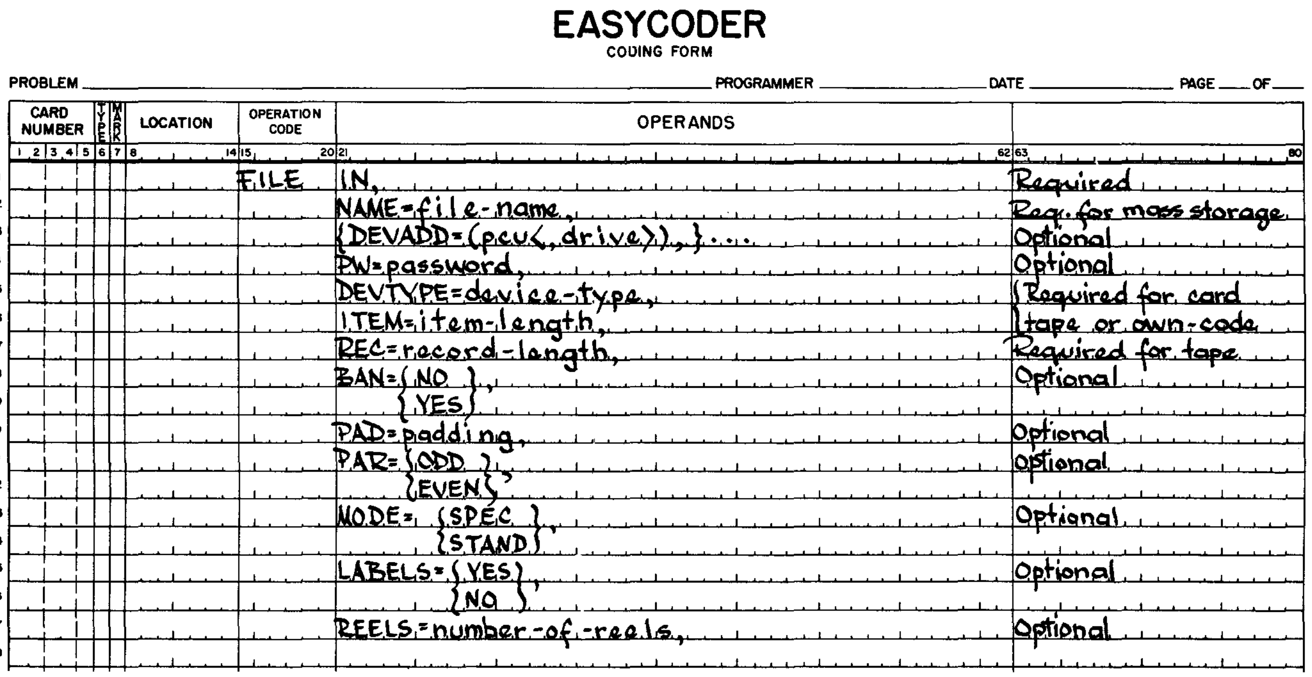
The File statements specify information about the input and output files used by the sort. The input/output function name, which is the first parameter of each File statement, specifies to which file the statement refers. The File statements are of four types:

1. Input (source-item) file;
2. Work file;
3. Output file; and
4. Information file.

Each type of File statement is described below under a separate heading.

INPUT FILE

The input file is the file of input data to the sort. The parameter "IN" identifies a File statement as referring to the input file (see Figure 4-4). The File statement for the input file must be present; there is no default assumption.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					FILE	IN,	Required
2						NAME=file-name,	Req. for mass storage
3						{DEVADD=(pcu<,drive>),}....	Optional
4						PW=password,	Optional
5						DEVTYPE=device-type,	{Required for card
6						ITEM=item-length,	{tape or own-code
7						REC=record-length,	Required for tape
8						BAN={NO },	Optional
9						{YES}	
10						PAD=padding,	Optional
11						PAR={ODD },	optional
12						{EVEN}	
13						MODE= {SPEC },	Optional
14						{STAND}	
15						LABELS={YES},	Optional
16						{NO }	
17						REELS=number-of-reels,	Optional
18							

Figure 4-4. File Statement and Parameters for Input File

Name Parameter

The name parameter (NAME) specifies the file name of the input file to the sort. The form of this parameter is shown below:

$$\boxed{\text{NAME = file-name,}}$$

<u>file-name</u> - The file name of the input file (up to ten characters in length). Trailing spaces are added.

For mass storage, the file name is required and is the name of the input file. For tape input, it is optional, and if specified, the file name of the input file will be checked against the value of the parameter (the default assumption is not to check the input file name). For card input and own-coding only, specifying this parameter has no meaning and is ignored.

Device-Type Parameter

The device-type parameter (DEVTYPE) specifies the storage medium used for the file as well as the type of peripheral device used to access the file. It may also be used to indicate that all input to the sort is to be fed through own-coding.

$$\boxed{\text{DEVTYPE = device-type,}}$$

<u>device-type</u> - The type number of the device used to access the file, or the common name of that device.

The device type may be chosen from the following list:

227	214-2
223	204B
224-1	NONE
224-2	

or if the common name is used:

CARD
TAPE

The parameter value must be NONE when all items to the sort are to be entered through own-coding.

The default assumption is mass storage. (Device type codes for mass storage are not used in the mass storage sort.)

Device-Address Parameter

This parameter (DEVADD) takes the following form.

```
{DEVADD = (pcu<, drive>) ,} ...
```

<u>pcu</u> - The peripheral address assignment of the control unit (two octal characters). Except for card input, the I/O bit is irrelevant but all other bits must be specified. If input is from cards, all six bits must be specified.

<u>drive</u> - Drive number (one octal character).

For mass storage, the device-address parameter specifies the physical device address(es) of the volume(s) containing the source item file. An input file may be contained on up to a maximum of eight volumes.

The default assumption is pcu 04, drive 0, and the file is contained on a single volume.

If the item address is appended, all volumes of the input file must be mounted on line and must, therefore, be specified in this parameter. If, however, the item address is not to be appended, the input file may be mounted on the minimum number of disk drives acceptable to its file organization. The sort will cycle through the device address assignments.

For tape, the default assumption is pcu 00, drive 1. Up to two devices may be specified, but the same pcu must be specified for both. For cards, the default assumption is 41. This parameter has no meaning for own-coding only.

For each device address being specified, DEVADD is necessary.

Password Parameter

The password parameter (PW) defines the password of the mass storage source-item file. The form of this parameter is shown below.

$\boxed{\text{PW = password,}}$

password - The source-item file password (up to eight characters in length). Trailing spaces are added.

The default assumption is that the file is not protected by a password.

Item Parameter

The item parameter (ITEM) specifies the number of characters in an item.

$\boxed{\text{ITEM = length-of-item,}}$

length-of-item - The size of the item in terms of the number of characters (up to four decimal characters).

This parameter is ignored if the input to the sort resides on mass storage. This parameter is required for all other input.

Record-Length Parameter

The record-length parameter (REC) specifies the number of characters in a record.

$\boxed{\text{REC = record-length,}}$

record-length - The number of characters in each record (up to four decimal characters). This includes a banner character when applicable.

This parameter is only significant if the input file is on tape. It should be omitted if the input to the sort comes from any other source.

Banner Parameter

This parameter (BAN) specifies whether or not the file is bannered and applies only to an input file on tape.

$$\boxed{\text{BAN} = \left\{ \begin{matrix} \text{NO} \\ \text{YES} \end{matrix} \right\} ,}$$

NO - The file is unbannered.

YES - The file is bannered.

The default value is YES.

Padding Character Parameter

This parameter (PAD) specifies a padding character for a magnetic tape file.

```
PAD = padding,
```

padding - The padding character expressed as two octal digits.

For a file with odd parity, the default value is octal 77; for a file with even parity, the default value is octal 11. If the first three characters of an item on tape are equal to the padding character, the item is bypassed.

Parity Parameter

This parameter (PAR) specifies the recording parity for a magnetic tape.

$$\boxed{\text{PAR} = \begin{Bmatrix}\text{ODD}\\ \text{EVEN}\end{Bmatrix},}$$

ODD - Odd parity.

EVEN - Even parity.

The default value is ODD.

Label Parameter

Magnetic tape files may have standard labels or no labels.

$$\boxed{\text{LABELS} = \begin{Bmatrix}\text{NO}\\ \text{YES}\end{Bmatrix},}$$

NO - The file has no labels and is terminated by a tape mark.

YES - The file has standard labels.

The default assumption is standard labels.

Reels Parameter

A magnetic tape file may be contained on more than one reel. This parameter (REELS) is required if there are no labels on a multivolume tape file.

```
REELS = number-of-reels,
```

number-of-reels - The number of reels of the file expressed as a single decimal digit.

The default assumption is a single reel. This parameter is ignored if the file has standard labels.

Mode Parameter

The mode parameter (MODE) specifies the card reading mode for a punched card file.

$$\boxed{\text{MODE} = \left\{\begin{array}{l}\text{STAND}\\ \text{SPEC}\end{array}\right\},}$$

STAND - The standard card mode.

SPEC - The special card mode.

The default value is special.

WORK FILES

The sort requires a work area in which it can create a sort item file on mass storage. This work area is assigned to the sort as one or more files. Normally, one file is sufficient for the work area. However, in some cases, one work file is not large enough to handle the sorting requirements. Therefore, up to five work files may be assigned. These files are referred to as work file 1, work file 2, etc. The parameter work$_n$ (where n is 1, 2, etc.) identifies the work file being referred to (see Figure 4-5).

If no File statement for work file 1 appears, the default assumption is that there is a work file named "*SORTWORK" on pcu 04, drive 0. The total number of units of allocation for all work files cannot exceed five.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1				FILE	WORKn	Optional
2					NAME=file-name,	Optional
3					{DEVADD=(pcu,drive),}...	Optional
4						

Figure 4-5. File Statement and Parameters for Work File

Name Parameter

The name parameter (NAME) specifies the name given to this work file. The form of this parameter is as follows.

$$\boxed{\text{NAME = file-name,}}$$

file-name - The file name of the work file (up to ten characters in length). Trailing spaces are added.

The default assumption is *SORTWORKΔ. DRIVE 0

Device-Address Parameter

The device-address parameter (DEVADD) specifies the physical device address(es) of the volume(s) containing the work file. The form of this parameter is shown below.

$\boxed{\left\{ \text{DEVADD = (pcu, drive),} \right\} \ldots}$

pcu - The peripheral address assignment of the control unit (two octal characters). The high-order bit is irrelevant, but all other bits must be specified.

drive - The drive number.

The default assumption is pcu 04, drive 0, and the work file is contained on a single volume.

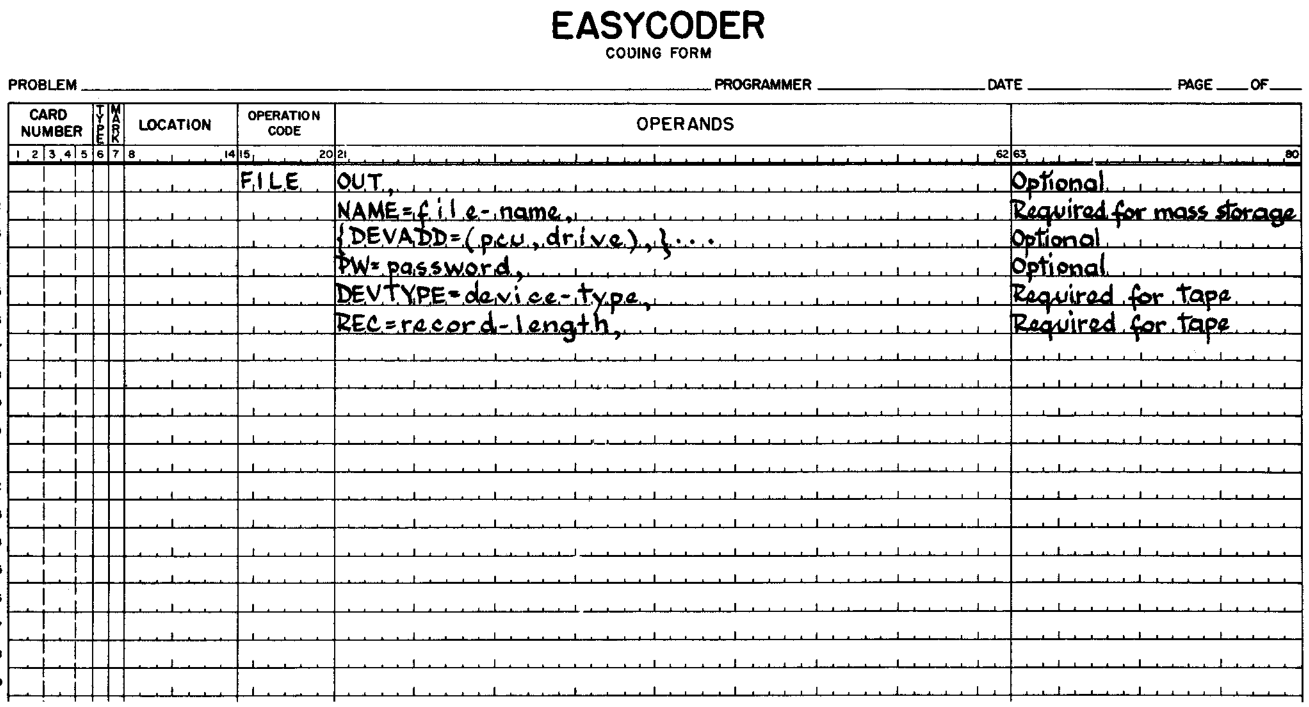
The total number of volumes that may be specified for all files is five, since the work area that can be utilized by the sort cannot exceed five units of allocation.

For each device address being specified, DEVADD is necessary.

OUTPUT FILE

If the File statement for the output file is present (see Figure 4-6), a request is made for the creation of a sort-ordered file either in a sequential file organization or on a 1/2-inch magnetic tape. At the completion of the sorting process, the sequential file program is executed.

The default assumption is that neither sequential file output nor tape output is required.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					FILE	OUT,	Optional
2						NAME=file-name,	Required for mass storage
3						{DEVADD=(pcu,drive),}...	Optional
4						PW=password,	Optional
5						DEVTYPE=device-type,	Required for tape
6						REC=record-length,	Required for tape
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							

Figure 4-6. File Statement and Parameters for Output File

Name Parameter

The name parameter (NAME) specifies the name of the sequential or tape output file. The form of the parameter is shown below.

```
NAME = file-name,
```

file-name - The name of the output file (up to ten alphanumeric characters).

For mass storage, a name must be specified for the sequential output file.

NOTE: It must be remembered that this file must be allocated prior to the execution of the sort.

If the file is on tape and it is to have standard labels, the value of the name parameter is the name given to the output file.

If the input to the sort was a tape file with standard labels, the default assumption is that no name change is required. If the input to the sort was from some other source, the default assumption is blanks.

Device-Type Parameter

The device-type parameter (DEVTYPE) specifies the storage medium to be used for the output file.

```
DEVTYPE = device-type,
```

device-type - The type number of the device used for the storing of the output file, or the common name of that device. The device type may be: 204B or TAPE.

The default assumption is mass storage (device-type codes for mass storage are not used).

Device-Address Parameter

The form of the device-address parameter (DEVADD) is shown below.

```
{DEVADD = (pcu, drive),}...
```

pcu - The peripheral address assignment of the control unit (two octal characters). The high-order (I/O) bit is irrelevant, but all other bits must be specified.

drive - Drive number (one octal character).

For mass storage, the device-address parameter(s) specifies the physical device address(es) of the volume(s) containing the output file. The output file may be contained on more than one volume, up to a maximum of eight. The default assumption is pcu 04, drive 0, and the file is contained on one volume.

For tape, up to two devices may be specified; however, the same pcu must be specified for both. The default assumption is pcu 00, drive 1.

For each address being specified, DEVADD is necessary.

Password Parameter

The password parameter (PW) defines the password of the mass storage sequential output file. This parameter takes the following form.

```
PW = password,
```

password - The sequential output file password (up to eight characters in length). Trailing spaces are added.

The default assumption is that the file is not protected by a password.

Record-Length Parameter

The record-length parameter (REC) indicates the number of characters in each record of the output file.

```
REC = record-length,
```

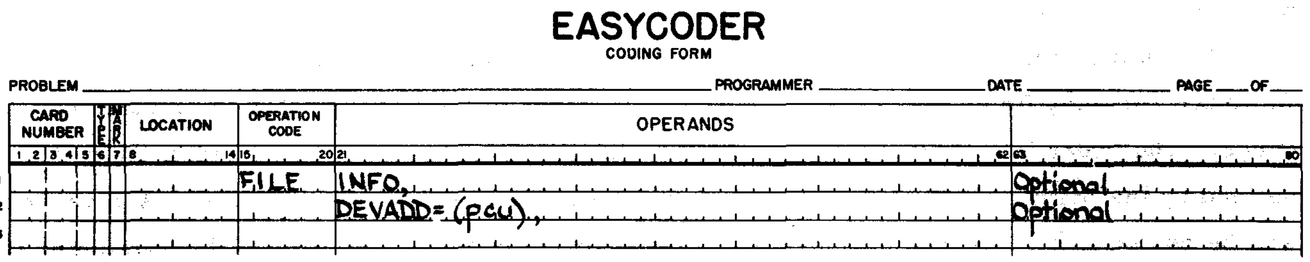
record-length - The number of characters in each record of the output file (up to four decimal characters). This includes the banner character when necessary.

This parameter is significant only if the output file is to be created on tape; in which case, it is required.

INFORMATION FILE

The information file is a printed program history. The parameter "INFO" identifies a File statement as referring to the information file (see Figure 4-7).

If no File statement for the information file appears, the default assumption is that no program history is required.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1				FILE	INFO,	Optional
2					DEVADD=(pcu),	Optional
3						

Figure 4-7. File Statement and Parameters for Information File

Device-Address Parameter

The device-address parameter (DEVADD) specifies the physical device address of the information file. The information file must be on a printer. The form of this parameter is as follows.

$$\boxed{\text{DEVADD = (pcu),}}$$

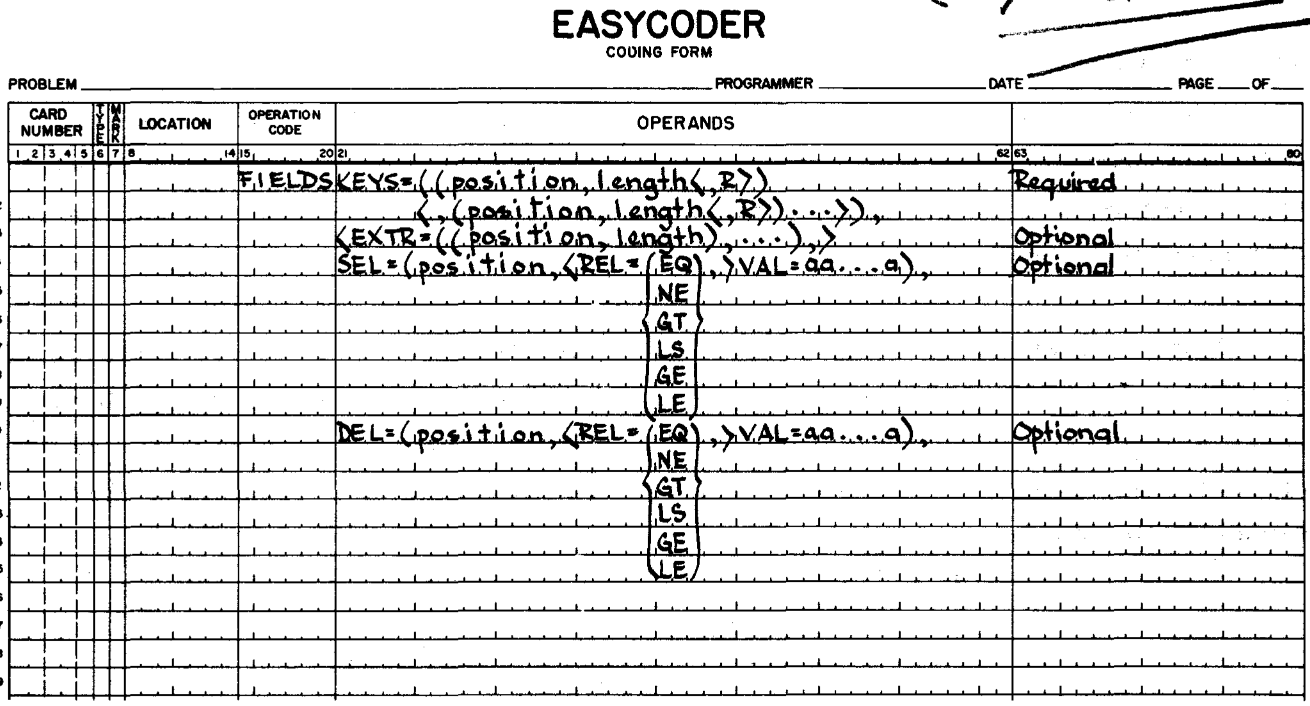
pcu - The peripheral address assignment of the control unit (two octal characters). The address must be in the first I/O sector.

The default assumption is pcu 02.

Fields Statement

Parameters belonging to the Fields statement (see Figure 4-8) define the functions of various fields of the source item in relation to the sort operation.

⟨ ⟩ = OPTIONAL



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1				FIELDS	KEYS=((position,length⟨,R⟩)	Required
2					⟨,(position,length⟨,R⟩),...⟩),	
3					⟨EXTR=((position,length),...),⟩	Optional
4					SEL=(position,⟨REL={EQ},⟩VAL=aa...a),	Optional
5					NE	
6					GT	
7					LS	
8					GE	
9					LE	
10					DEL=(position,⟨REL={EQ},⟩VAL=aa...a),	Optional
11					NE	
12					GT	
13					LS	
14					GE	
15					LE	
16						
17						
18						
19						

Figure 4-8. Parameters for the Fields Statement

1 Select / Sort

KEYS PARAMETER

The keys parameter (KEY or KEYS) specifies the sort-key data and is followed by a list. Up to ten sort-key fields may be specified. Their order in the list indicates their sort significance. The order is in decreasing importance. Keys are assumed to be sorted in the order defined by the primary sequence of the sort, unless a "reverse" key is specified. A reverse key will be sorted in the reverse sequence from the primary order. The form of the parameter is shown below.

```
KEYS = ( (position, length<,R>) . . . ),
```

<u>position</u>	- Up to four decimal digits giving the position of the leftmost (high-order) character of the key field in the item. The first character in the item is considered to be in position 0001.
<u>length</u>	- Up to two decimal digits giving the number of characters in the key field.
<u>R</u>	- If this parameter value is specified, the key with which it is associated is a reverse key. (When omitting this value, the comma preceding it is also omitted.)

The above three values (or two, if R is omitted) form a set which may be repeated up to nine times. Each set is enclosed in parentheses and separated by commas. Examples are shown below.

1. One key field - KEYS = (15, 4), (Note that with one key, only one pair of parentheses is needed.)
2. Three key fields with the second reverse sequence - KEYS = ((15, 4), (20, 5, R), (9, 1)),.

EXTRACT PARAMETER

The extract (EXTR) parameter specifies extract field information. Extract fields are fields of a source item that may be included in the sort item but have no significance in determining the final sort order. Up to 11 extract fields may be specified, and the order in the list determines their relative positions in the sort item.

When the user wishes the extract fields to be all those fields of the item that have not been specified as keys, then the value given to the extract parameter is "ITEM." The specification of ITEM implies that the output of the sort will be a string of items in the same format as the original source items. Key fields and extract fields must not overlap. The form of this parameter is as follows.

$$<\text{EXTR} = \left\{ \begin{array}{l} \text{((position, length), . . .)} \\ \text{ITEM} \end{array} \right\} , >$$

position	- Up to four decimal digits giving the position of the leftmost (high-order) character of the extract field in the item. The first character in the item is considered to be in position 0001.
length	- Up to two decimal digits giving the number of characters in an extract field.

The default assumption is that no extract fields are required.

SELECT PARAMETER

The select parameter (SEL) indicates that the sort application requires that only certain items of the input file be acceptable as input to the sort. The value of the select parameter is a list of two or three parameters (see format below). The first parameter (position) defines the position in the item of the field on which the selection is based. The second parameter (EQ, NE, GT, LS, GE, or LE), if specified, is preceded by a keyword (REL). The second parameter indicates the standard for selection, that is, equality or the degree of inequality with regard to the value specified by the third parameter, which is prefixed by the keyword VAL. This parameter defines the contents of the select field. Only one select parameter may be specified. The form of this parameter is shown below.

$$\boxed{\text{SEL = (position, < REL =}\left\{\begin{array}{c}\text{EQ}\\ \text{NE}\\ \text{GT}\\ \text{LS}\\ \text{GE}\\ \text{LE}\end{array}\right\}\text{, > VAL = aa...a),}}$$

position	- Up to four decimal digits giving the position of the leftmost (high-order) character of the field on which the selection is based. The first character in the item is considered to be in position 0001.
EQ	- equal
NE	- not equal
GT	- greater than
LS	- less than
GE	- greater than or equal
LE	- less than or equal
aa...a	- The value with which the defined field of an input item is to be compared. The maximum number of characters that the VAL parameter can define is 30. Blanks are significant.

If the second parameter in the list (prefixed by the keyword REL) is not specified, the standard for selection of an input item is equality with the value specified by the VAL parameter. The default assumption is that the select function is not required.

DELETE PARAMETER

The delete parameter (DEL) defines a field that allows an application of the sort to bypass those items which in the designated field bear a relationship to a specified value. The format of the delete parameter is the same as for the select parameter and is repeated here for the convenience of the reader. Only one delete parameter can be specified.

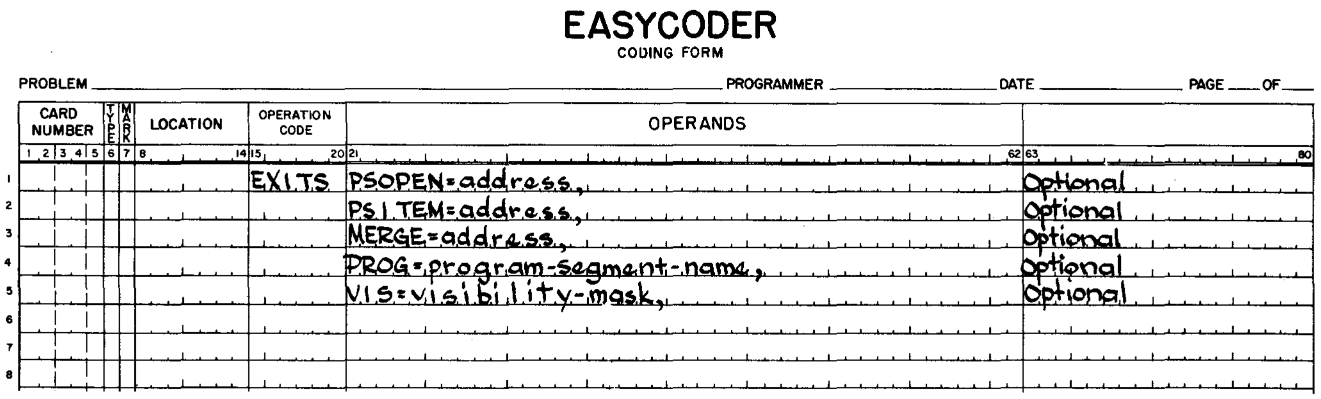
$$\boxed{\text{DEL} = (\text{position}, < \text{REL} = \left\{\begin{matrix}\text{EQ}\\ \text{NE}\\ \text{GT}\\ \text{LS}\\ \text{GE}\\ \text{LE}\end{matrix}\right\}, > \text{VAL} = \text{aa...a}),}$$

position	- Up to four decimal digits giving the position of the leftmost (high-order) character of the field on which the deletion is based. The first character in the item is considered to be in position 0001.
EQ	- equal
NE	- not equal
GT	- greater than
LS	- less than
GE	- greater than or equal
LE	- less than or equal
aa...a	- The value with which the defined field of an input item is to be compared. The maximum number of characters that the VAL parameter can define is 30. Blanks are significant.

If the second parameter in the list (prefixed by the keyword REL) is not specified, the standard for deletion of an input item is equality with the value specified by the VAL parameter. The default assumption is that the delete function is not required.

Exits Statement

The Exits statement (see Figure 4-9) specifies own-code exits from the sort to user procedures. If the Exits statement is omitted, there are no own-code exits.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					EXITS	PSOPEN=address,	Optional
2						PSITEM=address,	Optional
3						MERGE=address,	Optional
4						PROG=program-segment-name,	Optional
5						VIS=visibility-mask,	Optional
6							
7							
8							

Figure 4-9. Parameters for the Exits Statement

PRESORT OPEN EXIT PARAMETER

The presort open parameter (PSOPEN) specifies an own-code exit that permits the user to have access to the file description index item or to input tape labels. The default assumption is that there is no presort open exit. This parameter has the following form.

```
PSOPEN = address,
```

address - The address to which the sort will branch (up to six decimal digits in length).

PRESORT ITEM EXIT PARAMETER

This exit parameter (PSITEM) must be specified if all the items to be processed are to be entered through own-coding. If all input is not through own-coding, this parameter specifies an own-code exit that occurs for each accepted (selected) input item. The default assumption is that there is no presort item exit.

```
PSITEM = address,
```

address - The address to which the sort will branch (up to six decimal digits in length).

MERGE PARAMETER

The merge parameter (MERGE) specifies an own-code exit that occurs for each sort item when the final output of the sort is being created. The default assumption is that there is no exit. The merge own-code routine will be linked with the sort during the final phase of the merge. This parameter takes the following form.

```
MERGE = address,
```

address - The address to which the sort will branch (up to six decimal digits in length).

PROGRAM PARAMETER

The program parameter (PROG) specifies that a named program is to be loaded by the sort for the own-code merge exit. This parameter has the following form.

```
PROG = program-segment-name,
```

program-segment-name - The 8-character program-segment-name of the program to be loaded.

The default assumption is that the merge own-coding was resident in main memory before the final merge phase.

VISIBILITY PARAMETER

The visibility parameter (VIS) specifies the visibility mask of the merge own-code program. The visibility parameter has this form.

```
VIS = visibility-mask,
```

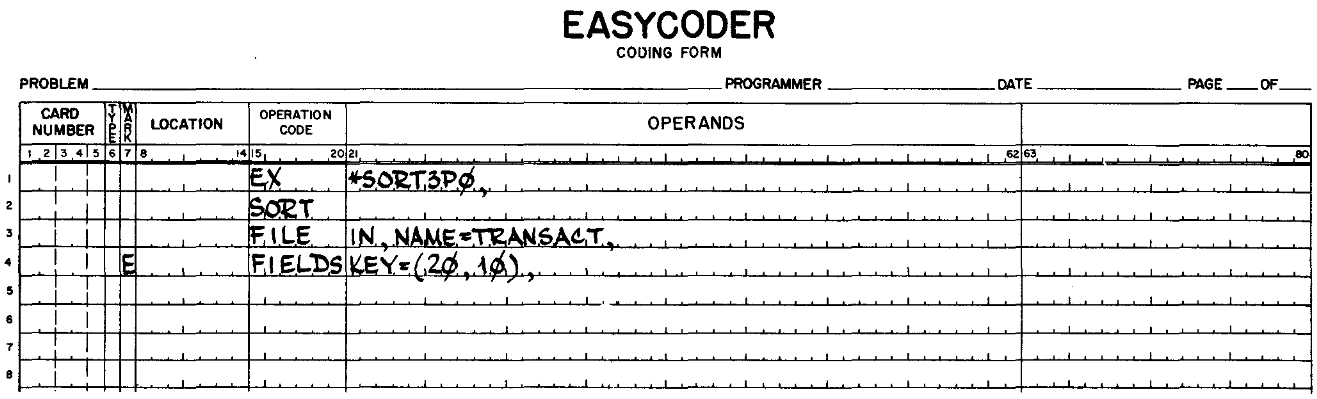
visibility-mask - The visibility mask to be used in loading the merge own-code program.

The default assumption is that visibility is not used when the sort is loading the merge own-code program.

Job Control Language Examples for Mass Storage Sort C

EXAMPLE 1

The source-item file is on mass storage and is named "TRANSACT" (see coded example below). The device address assigned is the assumed configuration 04, 0. One work file is available under the name "*SORTWORK," and its device address is likewise the assumed configuration 04, 0. The highest memory address is taken from the Supervisor communication area, and the output sequence is ascending. The sort item consists of one key with the hardware address of the associated input item appended.



EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE __ OF __

	CARD NUMBER 1-5	TYPE 6	MARK 7	LOCATION 8-14	OPERATION CODE 15-20	OPERANDS 21-62	63-80
1					EX	*SORT3P∅,	
2					SORT		
3					FILE	IN,NAME=TRANSACT,	
4			E		FIELDS	KEY=(2∅,1∅),	
5							
6							
7							
8							

EXAMPLE 2

A sort is to be executed on the file whose name is "TRANSACT" (see coded example below) and whose volume resides on a mass storage device with a peripheral address assignment of 07 for the control unit and a device number of 0. There are two work files with names and addresses given. The high memory address is 19,045, and the output is in descending sequence. The final output is a string of input items having the characters "LOBSTER△POT," starting in position 40.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					EX	*SORT3PØ,	
2					SORT	HMA=19Ø45, SEQ=D, ITADD=NO,	
3					FILE	IN, NAME=TRANSACT, DEVADD=(Ø7, Ø),	
4					FILE	WORK1, NAME=SPARE1, DEVADD=(Ø4, 1),	
5					FILE	WORK2, NAME=SPARE2, DEVADD=(Ø4, 2),	
6					FIELDS	KEYS=((3Ø,1Ø), (7Ø,6,R)), EXTR=ITEM,	
7			E			SEL=(4Ø, VAL=LOBSTERΔPOT),	
8							
9							

EXAMPLE 3

Sort the mass storage file "TRANSACT," but bypass as input to the sort those items that have the characters ZZZZ, beginning in position 10 (see coded example below). Two work files are available; the primary work file is named "*SORTWORK," and the secondary work file is named "EXWORK." Both are assigned the assumed device address (04, 0). The sort item consists of two extract fields followed by three key fields; the item address is appended. Merge own-coding is required, and the sort is to call the own-code program. The highest memory address is 8(4096)-1 = 32, 767.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					EX	*SORT3PØ,	
2					FILE	IN, NAME=TRANSACT, PW=ONLY ONE,	
3					FILE	WORK2, NAME=EXWORK,	
4					FIELDS	KEYS=((30,2), (1ØØ,5), (2Ø,4)), EXTR=((4Ø,1Ø),	
5						(9Ø,8)), DEL=(1Ø, VAL=ZZZZ),	
6					SORT	HMA=8M,	
7			E		EXITS	MERGE=29ØØØ, PROG=MERGOCØ1,	
8							
9							
10							

EXAMPLE 4

Sort the mass storage file that resides on two volumes. Three work files are available. The primary work file is named "*SORTWORK" and is on the volume assigned to control unit 04 and drive 0 (default assumptions). The second file is named "EXWORK" and is on the same volume as "*SORTWORK." "NEWWORK" is the name of the third work file, and it resides on two volumes. The output from the sort is to be the previously allocated sequential file "OUTSTAND"; it resides on two volumes.

EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					EX	*SORT3PØ,	
2					FILE	IN, NAME=TRANSACT, DEVADD=(Ø4,Ø), DEVADD=(Ø4,2),	
3						PW=ONLYONE,	
4					FILE	WORK2, NAME=EXWORK,	
5					FILE	WORK3, NAME=NEWWORK, DEVADD=(Ø4,2), DEVADD=(Ø4,3),	
6					FILE	OUT, NAME=OUTSTAND, DEVADD=(Ø4,3), DEVADD=(Ø4,4),	
7					FIELDS	KEYS=((3Ø,2),(1ØØ,5)),	
8			E		SORT	HMA=8M,	
9							
10							

EXAMPLE 5

The mass storage input file "TRANSACT" resides on the volume at control unit 04 and drive 0. The work file is "*SORTWORK" and resides on the same volume as the input file. The sort-item block size is to be restricted to four physical records, and a check is to be made to ensure that the size of the work area is sufficient to contain 5000 items. Select from the input file only those items with a 4-character field (starting in position 50) and with a value greater than or equal to 1000. The selected items are to be sorted on two keys in the commercial collating sequence.

EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					EX	*SORT3PØ,	
2					SORT	TRANS=YES, INVOL=5ØØØ, SIB=4,	
3					FILE	IN, NAME=TRANSACT,	
4					FIELDS	KEYS=((13Ø,2),(1ØØ,5)),	
5			E			SEL=(5Ø,REL=GE,VAL=1ØØØ),	
6							
7							
8							

EXAMPLE 6

The sort is not to be executed. Calculations are made to determine the sort-item block size and the work area compatibility. The results of these calculations are made available to the user by means of the printer.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
				EX	*SORT3PØ,	
				SORT	INVOL=5ØØØ,PRECAL=YES,	
				FILE	IN,NAME=TRANSACT,	
				FILE	WORK2,NAME=EXWORK,	
				FIELDS	KEYS=((3Ø,1Ø),(7Ø,6)),EXTR=ITEM,	
		E		FILE	INFO,DEVADD=(Ø2),	

EXAMPLE 7

In this example, the input file to the sort is on tape. The device address is pcu 00, drive 1. No name check is to be applied to the input file. Item length and record size are as specified, and the tape format is bannered, standard labels with odd parity. The padding character is 77_8. An output file on 1/2-inch tape is created; it contains the sort items. The output file name is "SORTED-OUT"; items on it are blocked two per record. The format of the output tape is the same as the input tape.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
				EX	*SORT3PØ,	
				SORT		
				FILE	IN,DEVTYPE=TAPE,ITEM=125,REC=5Ø1,	
				FILE	INFO,	
				FIELDS	KEYS=((3Ø,1Ø),(7Ø,6)),EXTR=ITEM,	
				FILE	OUT,NAME=SORTED-OUT,DEVTYPE=TAPE,	
		E			REC=251,DEVADD=(ØØ,2),	

EXAMPLE 8

All items to be processed are entered through own-coding. The sort branches to location 29,000 to receive each item and its corresponding mass storage address.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
				EX	*SORT3PØ,	
				SORT		
				FILE	IN,DEVTYPE=NONE,ITEM=1ØØ,	
				FIELDS	KEYS=((3Ø,1Ø),(7Ø,6)),	
				FILE	WORK1,NAME=EXWORK,	
		E		EXITS	PSITEM=29ØØØ,	

EXAMPLE 9

Input items are entered through the card reader at pcu 41. The standard mode is used for reading cards. Following the sort, the sequential output file program is called, and a file containing the sort items is created on 1/2-inch tape. The format of the output tape is bannered, standard labels in odd parity. Dummy labels are created and items are blocked two per record.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER 1-5	TYPE 6	MARK 7	LOCATION 8-14	OPERATION CODE 15-20	OPERANDS 21-62	63-80
1					EX	*SORT3DØ,	
2					SORT		
3					FILE	IN, DEVTYPE=223, ITEM=15Ø,	
4						DEVADD=(41), MODE=STAND,	
5					FIELDS	KEYS=((12,5),(9Ø,2Ø)), EXTR=ITEM,	
6					FILE	OUT, NAME=SORTED-OUT, DEVTYPE=2Ø4B,	
7			E			REC=3Ø1,	
8							
9							

<u>Examples of Work File Parameters Resident in Main Memory</u>

EXAMPLE 1

The file named "USEWORK" is to be used for the work area; it exists on two volumes which are to be assigned addresses (04, 0) and (04, 1). This work area must be described by two items in the parameter area, one for each volume as follows.

Characters	Values
51-60	USEWORKΔΔΔ
61-63	$(040000)_8$
64-73	USEWORKΔΔΔ
74-76	$(040100)_8$

EXAMPLE 2

Suppose that the file described in example 1 above is to be used, and a second work file is to be added. Assume that the second work file is called "DOESWORK" and that it resides on three volumes. The first volume of "DOESWORK" is the same as the first volume of "USEWORK," so that it is assigned address (04, 0). The second and third volumes of "DOESWORK" are separate from "USEWORK" and are to be assigned addresses (04, 2) and (04, 3). Then, the first two work-file fields (characters 51-76) remain as shown in example 1, and the following fields are defined.

Characters	Values
343-352	DOESWORKΔΔ
353-355	$(040000)_8$

Characters	Values
356-365	DOESWORK△△
366-368	$(040200)_8$
369-378	DOESWORK△△
379-381	$(040300)_8$

EXAMPLE 3

Three work files are available to the sort. The first of these is "*SORTWORK," and its three volumes are assigned addresses (04, 0), (04, 1), and (04, 2), respectively. "USEWORK" is a single-volume file on address (04, 1). The third work file "DOESWORK" exists across three volumes, but only one volume is to be used, and it is assigned address (04, 0). The parameter values in memory which describe the work area of the sort are as follows.

Characters	Values
51-60	*SORTWORK△
61-63	$(040000)_8$
64-73	*SORTWORK△
74-76	$(040100)_8$
343-352	*SORTWORK△
353-355	$(040200)_8$
356-365	USEWORK△△
366-368	$(040100)_8$
369-378	DOESWORK△
379-381	$(040000)_8$

Parameters Resident in Main Memory

When the parameters to the sort are resident in main memory prior to the sort being called, no default conditions are allowed; i.e., the complete parameter area must be specified. The starting location of the parameter area is 210 (decimal). Tables 4-2 and 4-3 provide summaries of the parameters resident in main memory.

If it is desired to calculate the decimal memory address of the parameter characters, add 209 (decimal) to the number in the "Characters" column of Tables 4-2 and 4-3.

Table 4-2. Sort Parameters Resident in Main Memory

Parameter Name	Characters	Value	Description
Input File[1] (Mass Storage Resident)	1-10	Input File Name	Source-item file name.
	11-18	Password	Password.
	19-21	XX_8 $0X_8$ 00_8	Address of primary input file control unit. Address of primary input file device. Address of primary input file pack address.
	22-24	Address of the second volume of a multivolume input file.	
	25-27 . . . 40-42	Address of the third volume of a multivolume input file. | Address of the eighth volume of a multivolume input file.	
Input File[1] (Tape Resident)	1-10	Input file name or blank.	
	11-18	Δ	Blank
	19-20	OX_8 XX_8 $0X_8$	Primary device address expressed in two characters: Tape control unit = XX_8 Drive = $0X_8$
	21	Δ	Blank
	22-23	Alternate device address or blank.	
	24	Δ	Blank

	25	
		Δ
		E
			Parity
			Δ = Odd
			E = Even

	26	
		N
		Δ
			Labels:
			N = No labels
			Δ = Standard labels

	27	Δ	Blank
	28	Padding character.	

	29	
		Δ
		N
			Banner parameter:
			Δ = Bannered file
			N = Unbannered file

	30	 T	Device type: T = Tape
	31	Number of reels or blank if standard labels.	
	32-42	Δ	Blank

Table 4-2 (cont). Sort Parameters Resident in Main Memory

Parameter Name	Characters	Value	Description
Input File[1] (Card)	1-10	Δ	Blank.
	11-18	Δ	Blank.
	19	XX_8	Peripheral control unit address.
	20-26	Δ	Blank.

	27	S
		Δ
			Mode:
			S = Standard card mode.
			Δ = Special card mode.

	28-29	Δ	Blank
	30	C	Device type: C = Cards.
	31-42	Δ	Blank
Input Through[1] Own-Coding	1-10		Blank
	11-29	Δ	Blank
	30	0	Device type: 0 = Own-coding (46 octal).
	31-42	Δ	Blank

Translation	43	Δ
		T
			Δ = No translation.
			T = Translation.

Precalculation Parameter	44	Δ Y	Δ = Sort is to be executed. Y = Sort is to return to the Supervisor after completing requested calculation.
Volume of Input Parameter	45-50		The number of input items in decimal with leading zeros. Blank - if no work area calculation is requested.
Work File[2] Parameters	51-60	Name of primary work file	Refer to the paragraph, "Examples of Work-File Parameters Resident in Main Memory" in this section, for handling multivolume work files.
	61-63	Device XX_8 $0X_8$ 00_8	Device address of primary work file expressed in three characters: XX_8 = Control unit, $0X_8$ = Device, and 00_8 = Magazine.
	64-73		Name of secondary work file. Blank if not used.
	74-76	Device	Device address of secondary work file expressed in three characters:

Table 4-2 (cont). Sort Parameters Resident in Main Memory

Parameter Name	Characters	Value	Description
Work File[2] Parameters (cont)		XX_8 $0X_8$ 00_8	XX_8 = Control unit $0X_8$ = Device 00_8 = Magazine Blank if not used.
Sort-Item Block Restriction Parameter	77-78		The maximum number of 250-character physical records that the sort can use to contain the sort-item block expressed in decimal with leading zeros. Blank if the option is not used.
Select and Delete Condition Parameter	79	See "Description" column.	Select parameter: 42 - for equal condition. 45 - for unequal condition. 41 - for greater than condition. 44 - for less than condition. 43 - for greater than or equal condition. 46 - for less than or equal condition. Blank if select option not used.
	80		Delete parameter: Values same as for select parameter. Blank if delete option is not used.
Information File	81	XX_8	Peripheral address assignment of printer control. Must be blank if not used. When this parameter is specified, the program history is printed.
Highest Memory Address	82-87	See "Description" column.	This is the highest memory address available to the sort and can be expressed in any of the following three ways: 1. In decimal, with leading zeros; 2. As the number of 4K modules, with leading zeros; or 3. As a binary address, right-justified in the parameter field and with leading spaces.
	88	Δ	Reserved for use of the operating system; must be set to blank.
Presort Own-Coding	89-94	See "Description" column.	The decimal address with leading zeros, to which the presort branches when a file description index or a tape label has been read. Blank if the option is not used. Alternatively, the address can be specified as a binary value, right-justified with leading blanks.
	95-100	See "Description" column.	The decimal address, with leading zeros, to which the presort will branch: (1) after the presort has been specialized, and (2) just

Table 4-2 (cont). Sort Parameters Resident in Main Memory

Parameter Name	Characters	Value	Description
Presort Own-Coding (cont)			prior to processing each item. Blank if the option is not used. Alternatively, the address may be specified as a binary value, right-justified with leading blanks. An address must be specified if input is through own-coding only.
Merge Own-Coding	101-106	See "Description" column.	The decimal address, with leading zeros, to which the final merge phase will branch when an item has been processed. The merge is said to be in its final phase when it is producing the single string output. Blank if this option is not used. Alternatively, the address may be specified as a binary value, right-justified with leading blanks.
Merge Own-Coding Program	107-112		Merge own-coding program name (this parameter is blank if the option is not used).
	113-114		Merge own-coding segment name.
	115-120		Visibility mask associated with merge own-coding program. Blank if not used.
Ascending or Descending Output	121	Δ	Ascending sequence. } When a mixed sequence is required, it is expressed through the reverse-key parameter.
		D	Descending sequence.
Item Address Appendage	122	Δ	Item address is to be appended to the sort-item.
		N	Item address is not to be appended to the sort-item.
Sort-Key Fields[3] (Up to ten key fields may be specified, in decreasing order of importance.)	123-126	dddd	Position of the primary sort key.
	127-128	dd	Number of characters in primary key.
	129	Δ	Key is to be sorted in the sequence defined by character 121.
		R	Key is to be sorted in sequence opposite that defined by character 121.
	130-136		Second key parameters (same as characters 123-129).
	137-143		Third key parameters (same as characters 123-129).
	144-150		Fourth key parameters (same as characters 123-129).
	151-157		Fifth key parameters (same as characters 123-129).

Table 4-2 (cont). Sort Parameters Resident in Main Memory

Parameter Name	Characters	Value	Description
Sort-Key Fields (cont)	158-164		Sixth key parameters (same as characters 123-129).
	165-171		Seventh key parameters (same as characters 123-129).
	172-178		Eighth key parameters (same as characters 123-129).
	179-185		Ninth key parameters (same as characters 123-129).
	186-192		Tenth key parameters (same as characters 123-129).
Extract Fields[4]	193-196	dddd ITEM	First extract field position. Extract fields are the residue of the item.
	197-198	dd	Number of characters in the first extract field.
	199-204		Second extract field (same as characters 193-198).
	205-210		Third extract field (same as characters 193-198).
	211-216		Fourth extract field (same as characters 193-198).
	217-222		Fifth extract field (same as characters 193-198).
	223-228		Sixth extract field (same as characters 193-198).
	229-234		Seventh extract field (same as characters 193-198).
	235-240		Eighth key extract field (same as characters 193-198).
	241-246		Ninth extract field (same as characters 193-198).
	247-252		Tenth extract field (same as characters 193-198).
	253-258		Eleventh extract field (same as characters 193-198).
Select Option	259-262	dddd ΔΔΔΔ	Position of the leftmost (high-order) character in the item of the field on which the sort is to be selected. Expressed in decimal with leading zeros. Select option not used.

Table 4-2 (cont). Sort Parameters Resident in Main Memory

Parameter Name	Characters	Value	Description
Select Option (cont)	263-293		These are the characters of the select field. Definition of the field must be terminated by a comma. Embedded blanks are significant. A comma may not be a character of the select field. The maximum size of a select field is 30 characters.
Delete Option	294-297	dddd ΔΔΔΔ	The position of the leftmost (high-order) character of the delete field of the item. Expressed in decimal with leading zeros. Delete option not used.
	298-328		These are the characters of the delete field. Definition of the delete field must be terminated by a comma. Embedded blanks are significant. A comma may not be a character of the delete field. The maximum size of the delete field is 30 characters.
Call Next Program	329-336		Program and segment name of the program to be loaded at the completion of the sort. If blank, the sort returns to the Supervisor at completion.
	337-342		Visibility mask associated with the program to be loaded. Blank if not used.
Additional Work File	343-352		Name of third work file.
	353-355	$(XX)_8$ $(0X)_8$ $(00)_8$	Address of third work file control unit. Address of third work file device. Address of third work file magazine.
	356-365		Name of the fourth work file.
	366-368		Device address of fourth work file.
	369-378		Name of fifth work file.
	379-381		Device address of fifth work file.
Output[5] File (Mass Storage Resident)	382-391		Name of the sequential output file, up to ten characters with trailing blanks. If blank, there is no request for a sequential output file.
	392-399		Password, if any, associated with the sequential output file. If blank, no password is required.
	400-402	 $(XX)_8$ $(0X)_8$ $(00)_8$	Device address for first volume (relative volume 0) of the file. Address of first volume control unit. Address of first volume device. Address of first volume magazine.

Table 4-2 (cont). Sort Parameters Resident in Main Memory

Parameter Name	Characters	Value	Description
Output File[5] (Mass Storage Resident) (cont)	403-405 421-423		Device address for second volume (relative volume 1) of the file. . . . Device address for eighth volume (relative volume 7) of the file.
Output File[5] (Tape Resident)	382-391		Name of output file, up to 10 characters with trailing blanks. Default assumption is blank. Blanks if there is no request for a tape resident output file.
	392-399	Δ	Blanks.
	400-401	XX_8 $0X_8$	Tape control unit. Drive. } Primary device address expressed as two characters.
	402		Blank.
	403-404		Alternate device address or blank.
	405-423		Blank.
Additional Input File Parameters	424-427		Input item length expressed in decimal with leading zeros. Must be blank if input file is on mass storage.
	428-431		Input record length expressed in decimal with leading zeros. Must be blank if input file is not on tape.
Additional Output File Parameters	432	T Δ	T = Tape. Δ = Mass Storage. } Output device type (also blank if there is no request for an output file).
	433-436		Output record length expressed in decimal with leading zeros. If blank, there is no request for a tape-resident output file.
Reserved by Sort	437-440	Δ	Reserved for future use by the sort (must be blank).

[1]Additional input file parameters appear in characters 424-431.

[2]Three examples of the use of work file parameters are provided on page 4-32.

[3]Specification of each field requires seven characters. Four characters are used for position of the leftmost (high-order) character in the field, and two are used for the number of characters in that field (both in decimal with leading zeros). The seventh character is defined as the reverse-key parameter, and its use permits the key with which it is associated to be in reverse sequence from that specified by the parameter in character 121. If less than ten key fields are used, the remaining key field parameter area must be blank. The first character of an item is considered to be in position 0001.

[4]If less than 11 extract fields are used, the remaining extract field parameter area must be blank.

[5]The application requiring that the final output of the sort be a sequential file that obeys the Data Management Subsystem conventions must describe the file in the output file parameters, unless the sequential output file program is to be called through the Execute statement.

Table 4-3. Summary of Sort Parameters Resident in Main Memory

Parameter Characters	Use
1-42 43 44 45-50	Input file parameters. Translation parameter. Precalculation parameter. Volume of input parameter.
51-60 61-63	Name of primary work file. Device address for primary work file.
64-73 74-76	Name of second work file. Device address for record work file.
77-78 79-80	Sort-item block restriction parameter. Select and delete condition parameter.
81	Printer control address assignment.
82-87	Highest memory address.
88	Reserved.
89-94	Presort open own-coding address.
95-100	Presort item own-coding address.
101-106 107-114 115-120	Merge item own-coding address. Merge own-coding program and segment name. Visibility mask.
121	Principal sort sequence.
122	Item address appendage.
123-129 186-192	First sort-key information. • • • Tenth sort-key information.
193-198 253-258	First extract field information. • • • Eleventh extract field information.
259-262 263-293	Select field position. Select field value.
294-297 298-328	Delete field position. Delete field value.
329-336	Program and segment name of segment to be loaded.
337-342	Visibility mask of program to be loaded.
343-352 353-355	Name of third work file. Device address of third work file.

Table 4-3 (cont). Summary of Sort Parameters Resident in Main Memory

Parameter Characters	Use
356-365 366-368	Name of fourth work file. Device address of fourth work file.
369-378 379-381	Name of fifth work file. Device address of fifth work file.
382-423	Output file parameters.
424-431	Additional input parameters.
432-436	Additional output file parameters.
437-440	Reserved for future use.

PROGRAMMER'S PREPARATION INFORMATION FOR MASS STORAGE SORT C

Work Files

Under certain conditions, the sort can utilize five work files. If five work files are provided for the sort work area, they may exist on the same volume, or each may be on a different volume. The device address associated with the five work files must be assigned to the same control unit. A work file must have been properly allocated through the File Support C program as a sequential file with 250-character records. All volumes of the work files must be mounted throughout the operation of the sort, the Fetch macro routine, and the sequential output file program.

All volumes of the work files must belong to the same device class (see Table 4-4). However, an input file does not have to belong to the same device class as its work files (see Table 4-5).

Table 4-4. Device Class Table

Device Class	Device Type
A	258, 259, 273, 259A, 259B
B	155
C	261, 262

Units of Allocation

The sort work area may be defined as five units of allocation or less. If the total number of units of allocation associated with the work files exceeds five, the extra units will be ignored, except that the sort uses the last record of the last unit of allocation for internal control information. This record is also used to store information for the Fetch macro routine. If the sequential output file program is requested for execution through sort parameters, the next to last record of the last unit of allocation is used to record information for the sequential output file program. When the number of units of allocation is five or less, the last two records of the last unit of allocation are similarly used.

Table 4-5. Work File Table

Input File	Work File
Class A	Class A or Class B or Class C
Class B	Class A or Class B or Class C
Class C	Class A or Class B or Class C

There are no restrictions on the units of allocation, but to achieve optimum efficiency, they should have the following characteristics.

1. The unit of allocation should consist of as few cylinders as possible.
2. The number of units of allocation should be as small as possible.
3. If a source file exists on one device and a work file is available on a second device, the work file should be specified to the sort as WORK1.
4. When the work file and the source file are on the same device, the work file should be as physically near the source file as possible.

Maximum Acceptable Sort-Item Size

The maximum sort-item size acceptable to the sort is the lesser of the two values derived by the following calculations. (The calculation of PS and PM in the following formulas is contained in Appendix A.)

1. $\frac{PS - INB - 25}{2}$

   INB = the size of the input block.

2. $\frac{PM - 34}{3}$

A further restriction on the size of the sort-item block is that it cannot exceed 3,741 characters in length.

Restriction of Sort-Item Block Size

The sort-item block size is determined by the sort so that its processing is as efficient as possible. It is possible that this sort-item block size might prove to be too large for the memory available during the execution of a process that includes the Fetch function. To avoid this difficulty, there is an option to allow the user to specify the maximum number of 250-character physical records that may be used to contain the sort-item block (see "Sort-Item Block Restriction Parameter," in this section). It is important that this option should only be used if the sort determined sort-item block size is too large to be used by some subsequent process.

Calculation of Required Sort Work Area

The sort handles up to five units of allocation that may be within one work file or split across as many as five work files. If the total number of units of allocation assigned to the file(s) is five or less, the last record of the last unit of allocation is used as storage for internal sort control information. It is also used to pass information to the Fetch function. If the sequential output file program is requested by the specification of a File Out statement in the sort job control file, the next to last record of the last unit of allocation is used to record information for the sequential output file program.

Refer to Appendix A for the calculation of some of the following parameters.

POSSIBLE NUMBER OF SORT ITEMS IN UNIT OF ALLOCATION

If a unit of allocation is defined as $C_1T_1C_2T_2$ where C_1T_1 stands for the numerical values of cylinder C_1 and track T_1, the number of 250-character physical records on a cylinder depends on the device type.

For Type 155, 258, 259, 273, 259A, and 259B Disk Pack Drives, the number of records per cylinder is determined by the formula:

$$15\ (T_2 - T_1 + 1) = NRC.$$

For a Type 261 or Type 262 Disk File, the number of records per cylinder is determined by the formula:

$$30\ (T_2 - T_1 + 1) = NRC.$$

After determining the number of records on a cylinder, proceed as follows:

1. Calculate the number of physical records required to contain a sort-item block (SIB): SIB = NI(SIS).

Using Table A-1 or A-2, find in column A the least value equal to or greater than SIB. The number of physical records required to contain a sort-item block is given by the corresponding value in column B. Let this value be represented by PRC. For example, if SIB = 730, a search of Table A-1 would show the value 741 in column A; therefore, PRC = 3 (the corresponding value in column B).

2. Calculate the number of sort items contained within a cylinder (NC):

$$NC = \left[\frac{NRC}{PRC}\right] (NI) \text{ (rounded to the next lowest integer).}$$

3. Calculate the number of sort items that may be contained within a unit of allocation:

   a. When the unit of allocation is not the last unit and the total number of units of allocation common to the specified work file(s) exceeds five, the number of sort items =

   $$NC \left[C_2 - C_1 + 1\right]$$

   b. When the unit of allocation is the last one to be used and the total number of units for the work file(s) is five or less, the number of sort items =

   $$NC \left[C_2 - C_1 + 1\right] - 2NI$$

   c. When the unit of allocation is the last one to be used and the total number of units of allocation common to the specified work file(s) exceeds five, the number of sort items =

   $$NC \left[C_2 - C_1 + 1\right] -NI$$

NUMBER OF WORK CYLINDERS REQUIRED TO HANDLE INPUT

If the difference between the highest and lowest tracks ($T_2 - T_1$) is the same for all units of allocation, the number of cylinders (CR) required to handle the input when there are five or less units of allocation is calculated as follows:

1. For Type 155, 258, 259, 273, 259A, and 259B Disk Pack Drives

$$CR = \frac{(I + 2NI)\ PRC}{15(T_2 - T_1 + 1)\ (NI)} \quad \text{(rounded up to the next highest integer).}$$

2. For Type 261 and 262 Disk Files

$$CR = \frac{(I + 2NI)\ PRC}{30(T_2 - T_1 + 1)\ (NI)}$$

I = total number of input items to the sort.

If the work files have more than five units of allocation, the formula becomes:

1. For Type 155, 258, 259, 273, 259A, and 259B Disk Pack Drives

$$CR = \frac{(PRC)\ (I + NI)}{15(T_2 - T_1 + 1)\ (NI)}$$

2. For Type 261 and 262 Disk Files

$$CR = \frac{(PRC)\ (I + NI)}{30(T_2 - T_1 + 1)\ (NI)}$$

Ensuring Sufficient Work Area

If the user provides, through the job control file, the size of the input file to the sort (refer to "Volume of Input Parameter," in this section) a calculation is made to ensure that the work area defined by the specification of the work files is sufficient for the successful completion of the sort. When the work area is insufficient, a message is given through the operator control file indicating the number of additional 10-track cylinders required to meet the demands of the input file. The message also gives the total number of 10-track work area cylinders required to contain all the items. When a printer is available, this information is also listed.

The Precalculation Option

The precalculation option (see "Precalculation Parameter" in this section) permits the calculation of both the sort-item block size and the work area compatibility. The information is made available through the printer and/or the operator control file and then a return is made to the Supervisor. The sort is not executed.

Sort Own-Coding

The various types of own-coding functions that may be performed during sort execution are described below. It is important that all own-coding routines which modify index registers restore them prior to returning control to the sort, with the exception of index registers X1 and X2 when they are used as the item linkage through which an item may be added. There are four types of sort own-coding:

1. Presort open;
2. Presort item-by-item;
3. Own-coding-only source of input to the sort; and
4. Merge own-coding.

The four types of sort own-coding are described below under separate headings.

The following definitions are given for the benefit of the reader.

1. Normal Return: A return of control to the sort from an own-code program which is made by branching to the location specified by the contents of the B-address register when control was given to the own-code program.
2. Branch-space Constant: The number of characters required for a branch operation code and one address field. If the sort is operating in the 3-character mode, the value of this constant is "4."

PRESORT OPEN OWN-CODING

Mass Storage Input

The presort branches to the location specified by the value of the presort open parameter when it has read the item in the file description index (*VOLDESCR*) of the first volume of the input file. There is an option not to exit to the open own-code program for every volume of a multivolume input file. Index register X1 contains the address of the left (high-order) end of that item. If accessing the input file requires the specifying of the password parameter, the item is not made available until this obligation has been met. Principally, the item can be inspected and moved to the own-code program area; no punctuation is present in the sort area it occupies. If the own-code program punctuates the sort area, it must clear that punctuation before returning control to the sort. If open own-coding is to be executed for successive volumes of the file, the presort is reentered at the location defined by the contents of the B-address register when the presort branched to the own-code program. Open own-coding can be terminated after any volume by branching to a location whose address is formed by adding a branchspace constant to the normal address. A store control register (SCR) instruction of the B-address register should be the first instruction of the presort open own-coding.

Tape Input (Standard Labels)

The presort branches to the location specified by the value of the presort open parameter when it has read the header from the first reel of the input file. There is an option not to exit to the open own-code routine for every reel of a multireel input file. Index register X1 contains the address of the left (high-order) end of that label. Principally, the label can be inspected and moved to the own-code program area; a word mark is present on the left end of the sort area it occupies. If the own-code program punctuates the sort area further, it must clear that punctuation before returning control to the sort. If open own-coding is to be executed for successive reel of the file, the presort is reentered at the location defined by the contents of the B-address register when the presort branched to the own-code program. Open own-coding can be terminated after any reel by branching to a location whose address is formed by adding a branchspace constant to the normal address. A store control register (SCR) instruction of the B-address register should be the first instruction of the presort open own-coding.

When the final trailer (1EOFΔ) of the input file has been read, the presort branches to the location specified by the value of the presort open parameter, unless the user has previously terminated open own-coding after examining a header. Index register X1 contains the address of the left (high-order) end of the label. An item mark is set on the location specified by the open address plus 1 to indicate the label is a trailer. Procedures followed by the own-code routine are identical to those used when the label is a header. The presort is reentered at the location defined by the contents of the B-address register when the presort branched to the own-code routine.

PRESORT ITEM-BY-ITEM OWN-CODING

The description of own-coding that follows assumes that own-coding is not the only source of input to the sort.

The five options in presort, item-by-item own-coding are described below under the following separate headings:

1. Processing an item;
2. Adding an item;
3. Deleting an item;
4. Terminating item own-coding; and
5. Terminating reading from the input file.

Processing an Item

The presort branches to the location specified by the value of the item-by-item parameter prior to processing each input item. However, if that item meets the standard for the Delete function, the item is not available to the user. Index register X1 contains the address of the left (high-order) character of the item as it resides in the sort's input buffer. The item's contents can be modified, but not its length. If the item is not from a mass storage index sequential file, a tape file, or a card file, punctuation is present in the item; a word mark appears on the high-order character of every key field and extract field pertinent to the particular sort application. Change in the status of the item punctuation leads to unpredictable results. Item punctuation is illustrated in Figure 4-10. Control is returned to the sort by making a normal return.

Adding an Item

An item, or those fields of an item relevant ot the sort application, can be added by placing the address of the left (high-order) character of the item in index register X1. The item must have the same format as the specified input, that is, item length, key fields, and so forth. Word marks must appear in the item in the following locations (unless the input file is a mass storage index sequential file, a tape file, or a card file in which no punctuation can be present in the item to be added):

1. In the first character of each key field and
2. In the first character of each extract field.

If an item sort has been specified, every nonkey field of the item is treated as an extract; thus, a word mark is required on the first character of all such fields. This convention is illustrated in Figure 4-10.

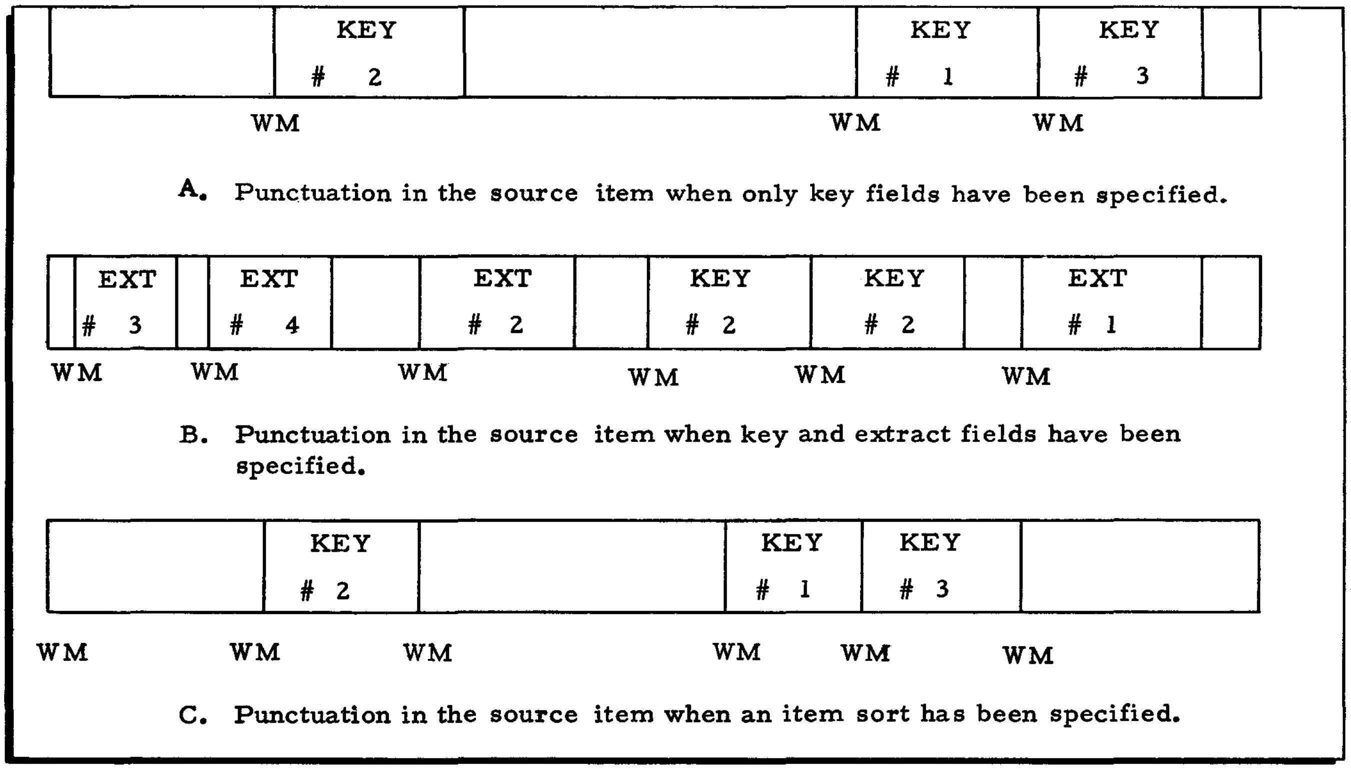

KEY # 2
KEY # 1
KEY # 3
WM
WM
WM
A. Punctuation in the source item when only key fields have been specified.
EXT # 3
EXT # 4
EXT # 2
KEY # 2
KEY # 2
EXT # 1
WM
WM
WM
WM
WM
WM
B. Punctuation in the source item when key and extract fields have been specified.
KEY # 2
KEY # 1
KEY # 3
WM
WM
WM
WM
WM
WM
C. Punctuation in the source item when an item sort has been specified.

Figure 4-10. Source-Item Punctuation

When the sort application specifies that an item address is to be appended to the sort item, the own-code program must supply this address to the sort. This is done by placing the address of the high-order character of an 11-character field in index register X2. The contents of this 11-character field can be an item hardware address, but the presence of a single character (77_8) in the high-order position of that field indicates to the Fetch macro routine and the sequential output file program that the item does not exist on the online mass storage device(s). When the address is a valid one, the format is as follows.

1.	Device Address	- Character 1:	Address assignment of control with high-order bit of zero.
		Character 2:	Drive number.
		Character 3:	Pack address.
2.	Block Address	- Characters 4 and 5:	Cylinder number.
		Characters 6 and 7:	Track number.
		Characters 8 and 9:	Number of record at the beginning of block containing the item.

Characters 10 and 11: Relative item number in the block; the number for the first item in the block is zero. For example, the fifth item in a block would have a relative item number of four.

No punctuation can be present in the 11-character field, except for an optional word mark in the high-order (leftmost) character. This field is illustrated in Figure 4-11.

PCU	D	P	C	C	T	T	R	R	I	I

where: PCU	=	Peripheral address assignment of control
D	=	Drive number
P	=	Pack address
CC	=	Cylinder
TT	=	Track
RR	=	Number of record at the beginning of block containing the item
II	=	Relative item number in the block, counting the first item in the block as zero

Figure 4-11. Contents of Item Address Appended to Sort Item

Control is returned to the sort by branching to a location whose address is formed by adding a branchspace constant to the normal return address.

Deleting an Item

When an item is to be deleted from the sort after inspection of that item by own-coding, the presort is reentered by branching to a location with an address formed by adding two branchspace constants to the normal return address.

Terminating Item Own-Coding

When all item own-code processing is completed, the own-coding routine must branch to a location whose address is obtained by adding three branch-space constants to the normal return address. Presort item own-coding can be terminated before or after the presort has processed all its input. However, the own-coding routine must be terminated. An item mark set on the location specified by the own-coding address indicates that the presort has processed all its input. If more items are to be added, the own-code program follows the procedure for adding an item. The presort will continue to exit to the own-code program until the terminating own-code return is made.

Terminating Reading from the Input File

The user can terminate reading from the input file by branching to a location, the address of which is obtained by adding four branchspace constants to the normal return address. The last item examined is not included in the input to the sort, and the presort make no further exits to the own-code routine.

OWN-CODING — ONLY SOURCE OF INPUT TO THE SORT

The presort branches to the location specified by the value of the PSITEM parameter in order to retrieve each input item. An item may be entered for processing by placing the address of the leftmost (high-order) character of the item in index register X1. No punctuation should appear in the item. When the sort application specifies that an item address (see "Item Address Parameter," in this section) is to be appended to the sort item, the own-code program must supply such an address to the sort by placing the address of the high-order character of an 11-character field in index register X2.

The contents of this 11-character field may reflect a genuine item hardware address. However, the presence of a single character $(77)_8$ in the high-order position of the field, indicates to the Fetch and sequential output file programs that the item does not reside on a mass storage device. When the address is a valid one, the format is as described in the paragraph, "Adding an Item," in this section.

No punctuation may be present in the 11-character field with the exception of an optional word mark on the high-order character.

Control is returned to the sort by branching to a location whose address is formed by adding a branchspace constant to the normal return address.

When all items have been entered, the own-coding routine must branch to a location whose address is obtained by adding three branchspace constants to the normal return address. It is necessary that own-code processing be terminated.

The presort continues to exit to the own-code program until the terminate own-coding return is made.

MERGE OWN-CODING

Only when the merge is creating its final 1-string output will it branch to the location specified by the merge own-coding parameter. The address of the high-order character of the sort item, as it is in the output buffer, will be found in index register X1 when the branch occurs. The item may be inspected, modified, or moved to the own-code area; but when control is returned to the sort, all punctuation of the item must be cleared to its original state. The punctuation and the format of the sort item, when the sort item is made available to the merge own-code program, are shown in Figure 4-12. When the merge has processed all the sort items, an item mark is set in the merge own-code location and an own-code branch is taken. It is not necessary that the program return control to the sort after this last exit is taken. Control is returned to the sort by making a normal return.

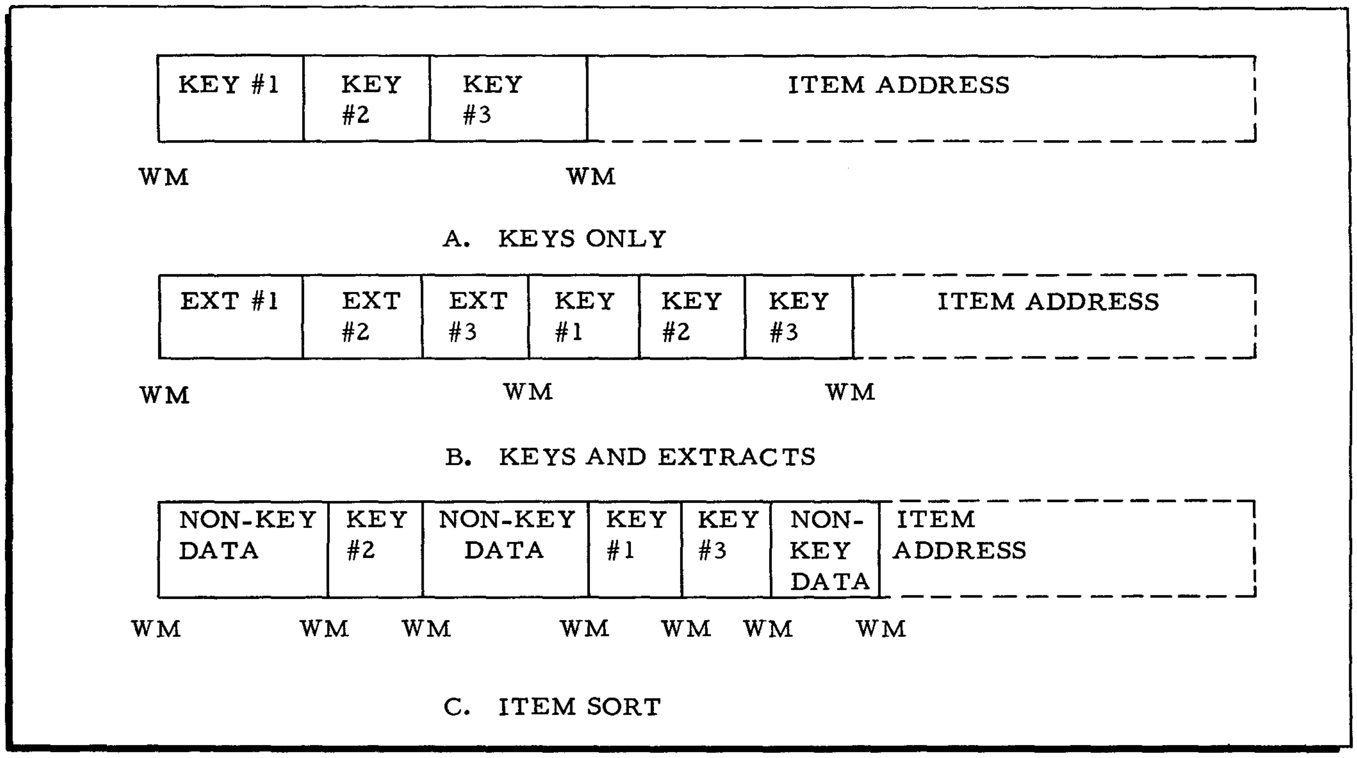


Figure 4-12. Punctuation and Format of Sort Item When Made Available to Merge Own-Code Program

DETAILED DESCRIPTION OF FETCH MACRO ROUTINE

The language elements and the programmer's preparation information for the Fetch macro routine are described below. Additionally, a description of the Fetch macro routine use of the Physical I/O C program is provided.

Language Elements of Fetch Macro Routine

The Fetch macro routine has the name "MFTCH." One Fetch macro call is required per program using the Fetch function. The format of the Fetch macro routine is shown below.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1		L		anytag	MFTCH	parameterØ1, ... parametern,	
2							
3							
4							

Table 4-6 provides the description of the Fetch macro parameters.

Table 4-6. Description of Fetch Macro Parameters

Parameter Number	Value
00	Any tag in the location field. Simply stated, this tag is the entrance point to which the user branches to initiate the Fetch. This parameter is required.
01	Two-character prefix applied to all symbols in the Fetch macro routine. This parameter is required.
02	Address for own-code exit (file open) when the Fetch macro routine has been fully specialized (symbolic tag or decimal address). This parameter is optional.
03	Address for own-code exit for sort-item examination (a symbolic tag or decimal address). This parameter is optional.
04	Address for own-code exit for source-item examination (symbolic tag or decimal address). This parameter is optional; however, at least one of the exits specified by parameters 03 and 04 must be specified. If this exit is to be utilized, the sort-item format specified to the preceding sort must include the source-item address.
05	Address for own-code exit when the end of the sort-item file is reached (symbolic tag or decimal address). This parameter is required.
06	Address of own-code exit for uncorrectable read error on sort-item file (symbolic tag or decimal address). This parameter is optional.
07	Address of own-code exit for uncorrectable read error on source-item file (symbolic tag or decimal address). This parameter is optional.
08	Buffer size required for reading blocks from the sort-item file (up to four decimal characters). Refer to OS in Appendix A. The default value is 1, 250.
09	Buffer mode for reading sort-item file: SINGLE - Single buffering. DOUBLE - Double buffering. The default value is "DOUBLE."
10	Index register used for sort-item file (one decimal character in the range from one to four). This index register will contain the address of the high-order character of the sort item at the own-code exit for each sort item. This parameter is required, because at least one of the two examination exits must be specified, and this parameter is used as a default value if parameter 13 is not written.
11	Buffer size required for reading blocks from the source-item file (up to four decimal characters). Refer to INB on page 4-43. The default value is 1, 250.
12	Buffering mode for reading source-item file: SINGLE - Single buffering. DOUBLE - Double buffering. The default value is "DOUBLE."

Table 4-6 (cont). Description of Fetch Macro Parameters

Parameter Number	Value
13	Index register used for source-item file (one decimal character in the range from one to four). This index register will contain the address of the high-order character of the source item at the own-code exit for each source item. The default value is the same index register as that specified by parameter 10.
14	Reserved for future use.
15	The address of a constant (DCW) in the user's program containing the name of the work file which holds the sort-item file input to the Fetch macro routine (symbolic tag or decimal address). The DCW must be ten characters long. When more than one work file is specified to the preceding sort, the name defined by the DCW must be that of the last work file of those specified. The default value is that the work file name is "*SORTWORK."
16	The address of a constant (DCW) in the user's program containing the device address of the device on which the work file named in parameter 15 can be found (symbolic tag or decimal address). The DCW must be two characters in length and contain the device address in exactly the same form it would appear in a PDT instruction. The default value is pcu 04, drive number 0.
17	A single alphanumeric character (non blank) that is the unique suffix to all tags in an MPIOC not called by this version of Fetch. If parameter 17 is blank, Fetch is used to issue a call for MPIOC.
18	Fetch operates in the partial processing mode if the value of this parameter is "P." Any other value results in normal mode processing. If the application of the Fetch macro routine is such that the examination of the sort item leads to bypassing of the source-item exit at least half of the time, then the partial processing mode saves processing time.

Programmer's Preparation Information for Using the Fetch Macro Routine

FETCH MACRO EXITS

Linkage between the user's program and any specialized version of Fetch that is embedded in the program is effected through Fetch exits. These exits are addresses in the user's program to which Fetch branches when it executes the function associated with a particular exit. (The addresses are specified through the parameters of the Fetch macro statement.) The two principal exits are:

1. Examine sort item; and
2. Examine source item (mass storage resident).

Examine Sort Item

When Fetch branches to the "sort-item" exit, a sort item has been brought into main memory, and the address of the left end of that item is available to the user through an index register that he has selected. The user may process that sort item and then return to Fetch to execute another request. Two returns are available: normal return and delete return. No punctuation is present in the sort item.

NORMAL RETURN: If the user wishes to access the mass-storage-resident source item that is associated with the current sort item, he executes a "normal return." This is done by branching to a location whose address was found in the contents of the B-address register when the "sort-item" exit was made. Fetch will then return to the user's program through the source-item exit with the main memory address of the requested item stored in a user-designated index register. However, if the source-item exit parameter was not specified to the Fetch macro routine, the next exit from Fetch will be effected through the sort item exit with the main memory address of the next logically sequential sort item. A source-item exit should not be specified if the source file is on 1/2-inch magnetic tape or cards.

DELETE RETURN: This return has meaning only if a source-item exit has been specified. When this return is used, it is assumed that the user is not interested in the source item associated with the current sort item. Therefore, the next exit from Fetch will be through the "sort-item" exit with the main memory address of the next logically sequential sort item.

The address of the "delete return" is equal to the "normal return" plus a constant equal to the length of the address mode plus one.

Examine Source Item

Fetch branches to the source-item exit when a mass-storage-resident source item is available in main memory. The main memory address of that item is given through a user-specified index register. Input items are at the disposal of the user and are presented in the sequential order of a particular sort application. After the user has processed an item, the return to Fetch is made by using the "normal return," as described above. No punctuation is present in the source item.

SUMMARY OF FETCH MACRO EXITS

There are two sets of exits: normal processing and error promoted.

1. <u>Normal-Processing Exits</u>
   a. File open (optional), normal return.
   b. Examine sort item (optional).

(1) Normal return.

(2) Delete return — inhibits the user's access to an associated mass storage resident source item where the "examine source item" is active.

c. Examine source item (optional), normal return.

d. End of Fetch processing (mandatory); <u>no return</u> is permitted.

2. <u>Error-Promoted Exits</u>

a. Uncorrectable read error on sort-item file (optional).

b. Uncorrectable read error on source-item file (optional).

Although the "examine sort-item" and "examine source-item" exits are classified as optional, Fetch requires that at least one of them be active.

MEMORY CONSIDERATIONS FOR USING THE FETCH MACRO ROUTINE

The maximum memory requirements for the Fetch macro routine, excluding requirements for buffer space and the Supervisor, is 5,500 locations. This requirement is reduced if all the options of the Fetch are not used, e.g., when no source-item examination exit is specified. The buffer space requirements depend upon parameter values; these values are discussed in the following paragraphs.

Examine Sort Item Only

If parameter 04 is blank, only the examine sort-item exit is specified. Single buffering is specified if parameter 09 is "SINGLE." In this case, buffer space = OS + 3, where OS is the sort-item block size. Double buffering is specified if parameter 09 is "DOUBLE" or if it is blank. In this case, buffer space = 2 (OS + 3).

Examine Source Item Only

If parameter 03 is blank, only the examine source-item exit is specified. Single buffering is specified if parameter 12 is "SINGLE." In this case, buffer space = (OS) + INB + 6, where INB is the source-item block size. Double buffering is specified if parameter 12 is "DOUBLE" or if the parameter value is left blank. In this case, buffer space = 2 (OS) + 2 (INB) + 12.

Both Source-Item and Sort-Item Exits Specified

When both the source-item and sort-item exits are specified, either file may be associated with single or double buffering. As a general rule, if there is a memory restriction, it is more advantageous to use single buffering for the sort item and double buffering for the source item. The following considerations apply.

1. If "SINGLE" is specified for both source-item and sort-item files, the buffer space required is OS + INB + 6.

2. If "DOUBLE" is specified for both source-item and sort-item files, the buffer space size required is 2 (OS + INB + 6).
3. If "SINGLE" (parameter 09) is specified for the sort-item file and "DOUBLE" (parameter 12) for the source-item file, the buffer space required is OS + 2 (INB) + 9.
4. If "DOUBLE" (parameter 09) is specified for the sort-item file and "SINGLE" (parameter 12) for the source-item file, the buffer space required is 2 (OS) + INB + 9.

Initiation of Fetch Macro Routine

The Fetch macro routine is initiated by branching to the tagged location specified in the location field (parameter 00) of the Fetch macro call.

Use of Physical I/O C Program by Fetch Macro Routine

The user of the Fetch macro routine has a choice in using the Physical I/O C program. He can use the Physical I/O C that can be incorporated (see parameter 17 in Table 4-6) in the Fetch macro routine, or he can have the Fetch macro routine use the Physical I/O C that is incorporated in the user's own-coding. This is illustrated in Figure 4-13.

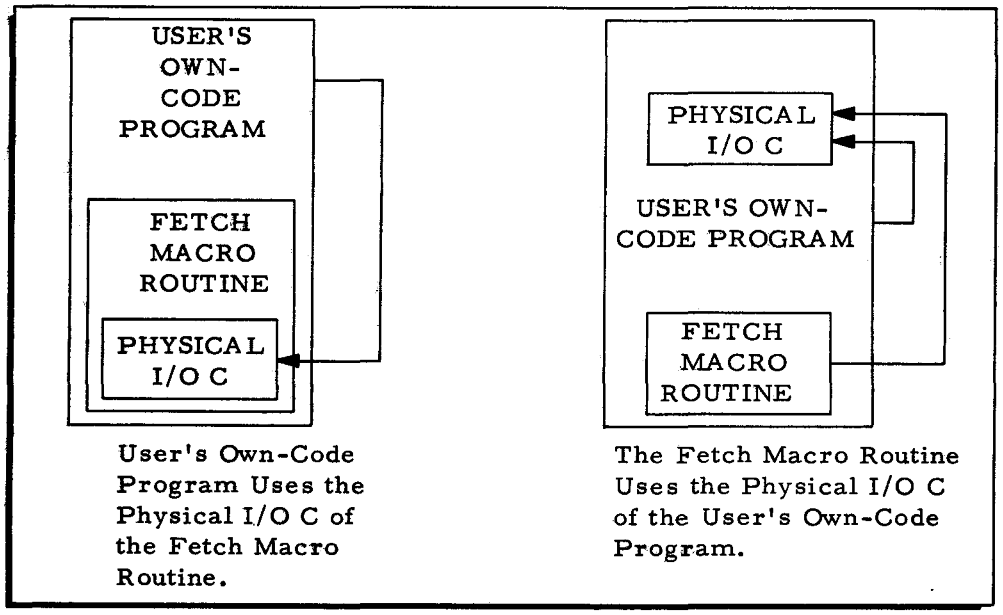

USER'S OWN-CODE PROGRAM
FETCH MACRO ROUTINE
PHYSICAL I/O C
User's Own-Code Program Uses the Physical I/O C of the Fetch Macro Routine.
PHYSICAL I/O C
USER'S OWN-CODE PROGRAM
FETCH MACRO ROUTINE
The Fetch Macro Routine Uses the Physical I/O C of the User's Own-Code Program.

Figure 4-13. Use of Physical I/O C by Fetch Macro Routine

In either case, the user should remember the requirements of Fetch for using Physical I/O C. Specifically, Fetch requires that parameter 04 of the MPIOC call be equal to "M" to permit control over more than one peripheral control unit. Parameters 02 and 05 should be blank.

In addition, the user may have more than one Fetch macro routine in a given program. The two methods of using Physical I/O C and the multiple occurrence of the Fetch macro routine in a given program are discussed in the following paragraphs.

OWN-CODE PROGRAM USES PHYSICAL I/O C OF FETCH MACRO ROUTINE

The Fetch macro routine incorporates Physical I/O C automatically if parameter 17 of the macro call for Fetch is blank. If the user intends to use Logical I/O C or to use Physical I/O C directly in his own-coding, he should use Physical I/O C as provided by the Fetch macro routine. The following points should be noted.

1. The MPIOC suffix associated with Physical I/O C when called by the Fetch macro routine is % (+, 8, 5 keypunch). This parameter must be supplied to Logical I/O C or to the Physical I/O C that the own-code program uses.
2. The file tag used by Physical I/O C when called by the Fetch macro routine is the value of the parameter 01 of the Fetch macro call. The user should not use tags beginning with this file tag in his own-code. If Logical I/O C or Physical I/O C is used in an own-code program, this same file tag should not be assigned to these input/output routines.

FETCH MACRO ROUTINE USES PHYSICAL I/O C OF OWN-CODE PROGRAM

If parameter 17 of the macro call to the Fetch is not blank, the Fetch macro uses the Physical I/O C supplied within the user's own-code program. The value of parameter 17 is the MPIOC suffix associated with that particular Physical I/O C. The following considerations should be noted.

1. The MPIOC suffix supplied to the Physical I/O C that Fetch is to use must be the value of parameter 17 of the Fetch macro call.
2. The file tag supplied in parameter 01 of the Fetch macro call must be different from any file tag that is used with Logical I/O C or Physical I/O C in the own-code program.

MULTIPLE OCCURRENCE OF FETCH MACRO ROUTINE IN PROGRAM

If the user desires to have more than one Fetch macro routine in a single program, this can be accomplished either with Physical I/O C being called by the Fetch macro or with the Fetch macro routine using Physical I/O C as supplied by the user's own-code program. The following considerations apply.

1. If Physical I/O C is to be called by Fetch, the first macro call for Fetch must leave parameter 17 blank, and all other calls for Fetch must specify "%" for parameter 17.
2. If Fetch is to use the Physical I/O C supplied with the own-code program, then every call for the Fetch macro routine must specify the value of parameter 17 as the suffix associated with the MPIOC to be used by the Fetch macro routine.

In both cases described above, each call for the Fetch routine must specify a different value for parameter 01 (the Physical I/O C file tag).

PARTIAL PROCESSING

If the Fetch macro routine has been specialized for the partial processing mode (parameter 18 is "P"), the buffering modes for the sort-item file and the source-item file are both single. The Fetch macro routine ignores the values of parameters 9 and 12 and in both instances, single buffering is forced. Partial processing should only be used if examination of the sort item leads to a limited accessing of the source-item file.

DELAYED EXECUTION

If the Fetch macro routine is not used immediately following the sort, it is essential upon execution of the program containing the Fetch routine that the source-item file and the work files have the same assignments that were given to them at the beginning of the sort.

DETAILED DESCRIPTION OF SEQUENTIAL OUTPUT FILE PROGRAM

Job Control Language for Sequential Output File Program

Requesting the creation of a sequential output file through a separate Execute statement (with parameters submitted by means of other job control statements) allows the use of own-coding during the operation of the sequential output file program. Own-coding is not available if the sort calls for the sequential output file program through the inclusion of the File statement for the output file.

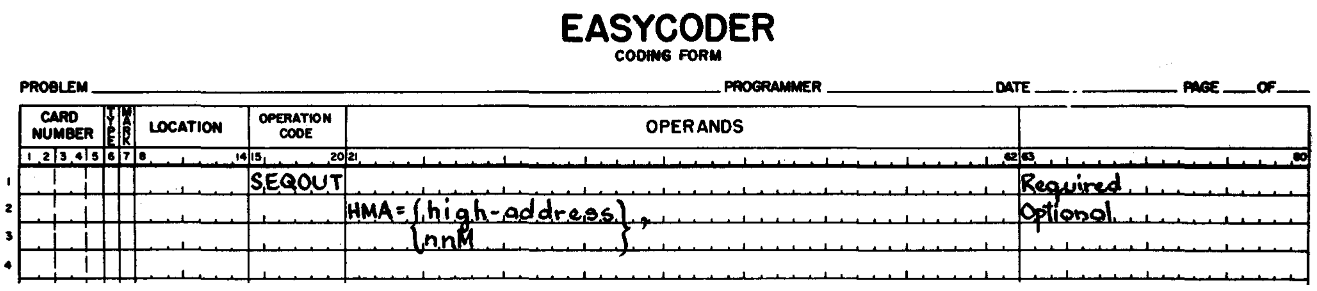
The following paragraphs describe the job control language when the sequential output file program is called through its own Execute statement. The job control language for the case where the sequential output file program is called by the sort is described in this section under "Job Control Language for Mass Storage Sort C."

Execute Statement

An Execute statement is necessary to call the sequential output file program. This Execute statement must have "*MSEQFS0" in the operands field.

Sequential Output Statement

The Sequential Output statement is required. The format of this statement is shown in Figure 4-14.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1				SEQOUT		Required
2					HMA= {high-address},	Optional
3					{nnM}	
4						

Figure 4-14. Sequential Output File Program Statement and Parameter

HIGH-MEMORY-ADDRESS PARAMETER

The high-memory-address parameter (HMA) defines the highest memory address available to the sequential output file program. If an own-code program is resident during the program and the base address of that program is lower than that specified for the HMA parameter, then the base address (lower exit address) of that program less one is taken as being the highest memory address available for the execution of the sequential output file program. The form of this parameter is shown below.

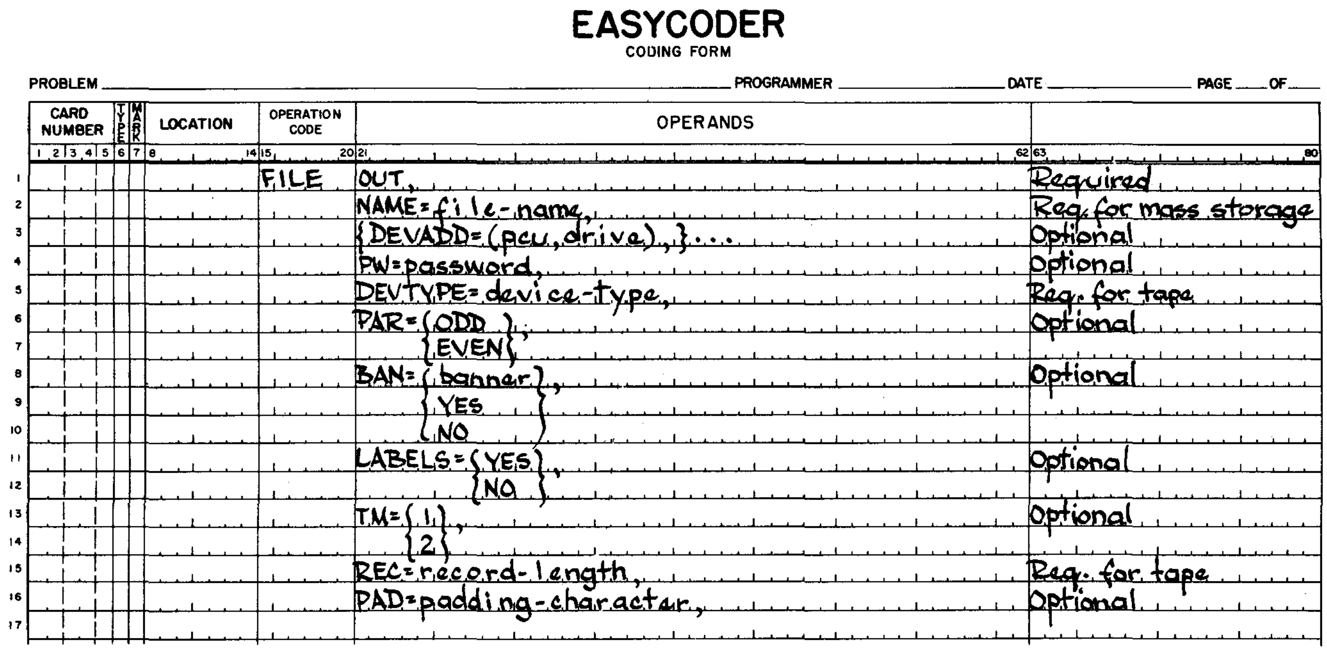
$$\boxed{\text{HMA} = \left\{\begin{matrix}\text{high-address}\\ \text{nnM}\end{matrix}\right\},}$$

high-address	- The highest address available, written as a decimal number (up to six characters in length).
nn M	- The number of memory modules (M) available, written as a 2-character decimal number. The highest address available is computed by the sequential output file program and is one less than nn times 4,096. "M" is a parameter constant.

The default assumption is that the HMA value is to be taken from the Supervisor communication area field that specifies the highest memory address available to the object program. If HMA is specified but has a higher value than the address in the Supervisor, the Supervisor value is used.

File Statements

The File statements specify information about the input and output files used by the sequential output file program. The format of the File statements is shown in Figure 4-15.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1					FILE	OUT,	Required
2						NAME=file-name,	Req. for mass storage
3						{DEVADD=(pcu,drive),}...	Optional
4						PW=password,	Optional
5						DEVTYPE=device-type,	Req. for tape
6						PAR={ODD },	Optional
7						{EVEN}	
8						BAN={banner},	Optional
9						{YES}	
10						{NO}	
11						LABELS={YES},	Optional
12						{NO}	
13						TM={1},	Optional
14						{2}	
15						REC=record-length,	Req. for tape
16						PAD=padding-charactar,	Optional
17							

Figure 4-15. File Statement and Parameters for Output File

OUTPUT FILE

A sequential file is the output of the sequential output file program. The parameter "OUT" identifies a File statement as referring to an output file. This statement is required.

Name Parameter

The name parameter (NAME) specifies the file name of the output file. It has the following form.

```
NAME = file-name,
```

file-name - The file-name of the output file (up to ten characters in length). Trailing spaces are added.

For mass storage, the file-name is the name of the output file and is a required parameter. For magnetic tape, if the output file has standard labels, this file-name should be given to the file. The output file-name is blank if this parameter is not specified.

Device-Address Parameter

The device-address parameter (DEVADD) specifies the physical device address(es) of the volume(s) containing the output file. It has the following format.

```
{DEVADD = (pcu, drive),} ...
```

pcu - The peripheral address assignment of the control unit (two octal characters). The high-order (I/O) bit is not relevant, but all other bits must be specified.

drive - Drive number (one octal character).

For mass storage, an output file may be contained on more than one volume, up to a maximum of eight. The default assumption is pcu 04, drive 0, and that the file is contained within one volume.

For magnetic tape, no more than two devices may be specified, and both devices must have the same pcu number. The default assumption is pcu 00, device 1.

Password Parameter

The password parameter (PW) defines the password of the sequential output file. It has the following format.

$$\boxed{\text{PW = password,}}$$

password - The output file password (up to eight characters). Trailing spaces are added.

The default assumption is that the file is not protected by a password.

Device-Type Parameter

The device-type parameter (DEVTYPE) specifies the storage medium to be used for the output file, and it has the following format.

$$\boxed{\text{DEVTYPE = device-type,}}$$

device-type - The type number of the device used for storing the output file or the general name of that device. The device type may be 204B or TAPE.

The default assumption is mass storage (device type codes for mass storage are not used).

Parity Parameter

The parity parameter (PAR) specifies the parity of recording for a magnetic tape file and has the following format.

$$\boxed{\text{PAR} = \left\{\begin{matrix}\text{ODD}\\ \text{EVEN}\end{matrix}\right\},}$$

ODD - Odd parity.

EVEN - Even parity.

The default value is ODD.

Banner Parameter

The banner parameter (BAN) specifies information concerning the banner character, and it has the following format:

$$\boxed{\text{BAN} = \begin{Bmatrix}\text{banner}\\ \text{YES}\\ \text{NO}\end{Bmatrix},}$$

<u>banner</u> - A banner character is to be the first character of every record (specified as two octal digits).

<u>YES</u> - A banner character of 56 octal is to be the first character of every record.

<u>NO</u> - No banner is required.

The default assumption is YES. Specifying the banner parameter has no significance if the output file is to reside on mass storage.

Label Parameter

The label parameter (LABELS) specifies whether the magnetic tape file is to have standard labels or no labels. The parameter has the following format.

$$\boxed{\text{LABELS} = \begin{Bmatrix}\text{YES}\\ \text{NO}\end{Bmatrix},}$$

<u>YES</u> - The file has standard labels.

<u>NO</u> - The file has no labels and is terminated by a tape mark.

The default assumption is standard labels.

Tape-Mark Parameter

The tape-mark parameter (TM) describes the tape-mark requirement when standard labels are requested.

$$\boxed{\text{TM} = \begin{Bmatrix}1\\ 2\end{Bmatrix},}$$

<u>1</u> - A tape mark precedes the standard trailer.

<u>2</u> - A tape mark follows the standard header and precedes the standard trailer.

The default assumption is that tape marks are not required with standard labels. The tape-mark parameter is ignored if labels are absent. All tape marks will be written in even parity marks.

Record-Length Parameter

The record-length parameter (REC) indicates the number of characters in each record of the output file. It is of the following form:

$$\boxed{\text{REC = record-length,}}$$

<u>record-length</u> - The number of characters in each record of the file including a banner character when applicable (up to four decimal characters).

This parameter is significant only if the output file is on magnetic tape.

Padding Character Parameter

The padding character parameter (PAD) specifies a padding character for a magnetic tape file, and it has the following format.

$$\boxed{\text{PAD = padding,}}$$

<u>padding</u> - Padding character expressed as two octal digits.

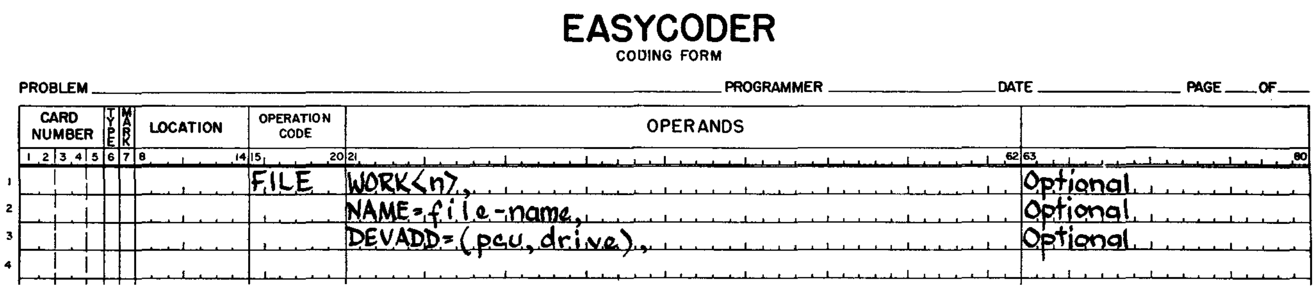
For an odd parity file, the default value is 77_8; for an even parity file, the default value is 11_8.

WORK FILE

A work file is a file in which the preceding sort used a work area on mass storage which was defined by work file parameters to that sort. A parameter record essential to the execution of the sequential output file program has been written in the mass storage area allocated to the last work file specified to the preceding sort. The name of this work file, therefore, is required by the sequential output file program, together with the device associated with the last volume of the work file.

The parameter "WORK<n>" identifies the file statement as referring to a work file (see Figure 4-16).

If no File statement appears for the work file, the default assumption is that only one work file was specified to the sort; its name is "*SORTWORK," and it is assigned device address pcu 04, drive 0. If the file statement is given, it is not necessary that the "n" of WORK<n> be specified.



EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE __ OF __

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					FILE	WORK<n>,	Optional
2						NAME=file-name,	Optional
3						DEVADD=(pcu, drive),	Optional
4							

Figure 4-16. File Statement and Parameters for Work File

Name Parameter

The name parameter (NAME) specifies the file name of the last work file. This parameter is required. It has the following form.

$\boxed{\text{NAME = file-name,}}$

file-name - The file-name of the work file (up to 10 characters in length). Trailing spaces are added.

The default assumption is *SORTWORK.

Device-Address Parameter

The device-address parameter (DEVADD) specifies the physical device address of the last volume of the last work file. This parameter has the following form.

$\boxed{\text{DEVADD = (pcu, drive),}}$

pcu - The peripheral address assignment of the control unit (two octal characters). The high-order (I/O) bit is not relevant, but all other bits must be specified.

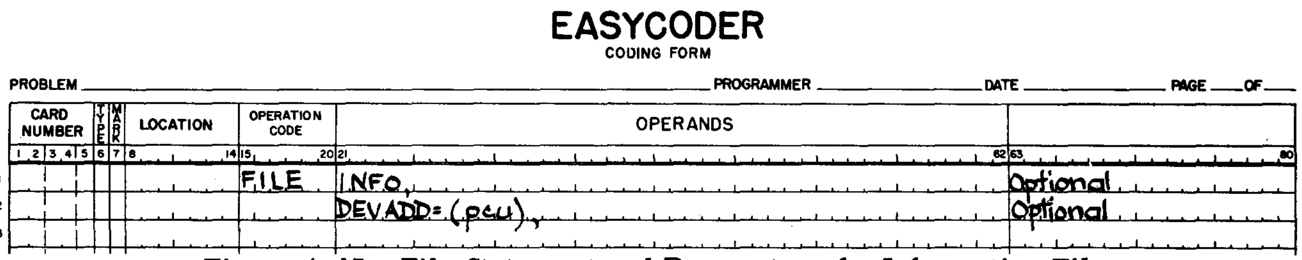
drive - Drive number (one octal character).

The default assumption is pcu 04, drive 0.

INFORMATION FILE

The information file is a printed program history. The parameter "INFO" identifies a file as referring to the information file (see Figure 4-17).

If no File statement for the information file appears, the default assumption is that a program history is not required.



EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE __ OF __

CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1				FILE	INFO,	Optional
2					DEVADD=(pcu),	Optional
3						

Figure 4-17. File Statement and Parameters for Information File

Device-Address Parameter

The device-address parameter (DEVADD) specifies the physical device address of the information file. The information file must be on the printer. The device-address parameter has the following form.

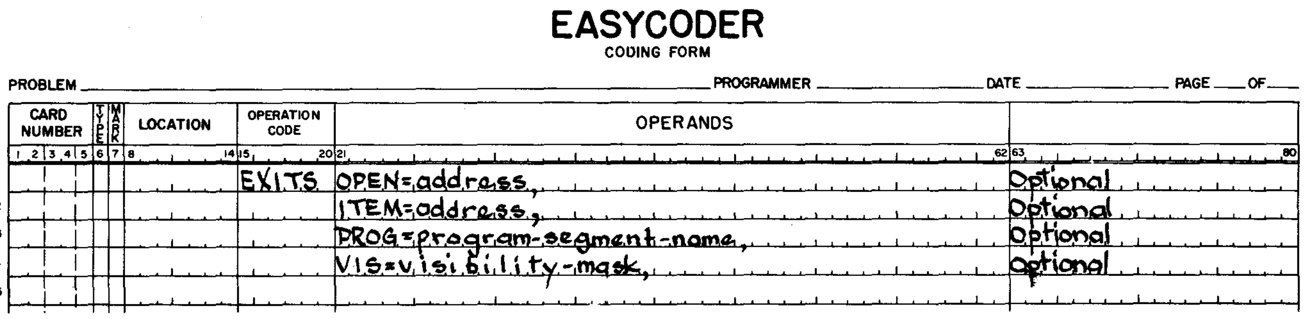
DEVADD = (pcu),

pcu - The peripheral address assignment of the control unit (two octal characters).

The default assumption is pcu 02.

Exits Statement

The Exits statement (see Figure 4-18) specifies own-code exits from the sequential output file program to user routines. There are no own-coding exits if the Exits statement is omitted. For proper programming procedures, refer to the heading "Programmer's Preparation Information for Sequential Output File Program," in this section.



EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE __ OF __

CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1				EXITS	OPEN=address,	Optional
2					ITEM=address,	Optional
3					PROG=program-segment-name,	Optional
4					VIS=visibility-mask,	Optional
5						

Figure 4-18. Exits Statement and Parameters

OPEN PARAMETER

If the output file is on a mass storage device, the open parameter (OPEN) specifies an own-code exit that permits the user to have access to the item in the file description index pertinent to the output file. If the output file is to be created on 1/2-inch tape with standard labels, the open parameter specifies an own-code exit that permits the user to access the labels before they are written on tape.

```
OPEN = address,
```

address - The address to which the sequential output file program will branch (up to six decimal digits).

The default assumption is that there is no OPEN exit.

ITEM PARAMETER

The item parameter (ITEM) specifies an own-code exit that occurs for every sort item until the user terminates the function. The default assumption is that there is no exit. The form of this parameter is shown below.

```
ITEM = address,
```

address - The address to which the sequential output file program will branch (up to six decimal digits).

PROGRAM NAME PARAMETER

The program name parameter (PROG) specifies that a named program is to be loaded for the item and open own-code exits. This parameter has the following form.

```
PROG = program-segment-name,
```

program-segment-name - The program segment name of the program to be loaded.

The default assumption is that the own-code program was resident in main memory before the sequential output file program was loaded.

VISIBILITY PARAMETER

The visibility parameter (VIS) specifies the visibility mask of the own-code program. This parameter has the following form.

```
VIS = visibility-mask,
```

visibility-mask - The 6-character visibility mask to be used in loading the own-code program.

The default assumption is that visibility is not used.

Job Control Language Examples for Sequential Output File Program

EXAMPLE 1

This example illustrates a call for a sort followed by a request to create a sequential output file based on output from that sort. The work-file information corresponds to that defined for the last volume available to the sort for a work area. Own-coding is requested during the creation of the output file, and the own-code program is to be called by the sequential output program.

EASYCODER
CODING FORM

PROBLEM ________ PROGRAMMER ________ DATE ________ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					EX	*SORT3PØ,	
2					FILE	IN, NAME=TRANSACT,	
3					FILE	WORK1, NAME=WORKOUT, DEVADD=(Ø4,1),	
4					FILE	WORK2, NAME=EXWORK, DEVADD=(Ø4,1),	
5						DEVADD=(Ø4,2),	
6					FIELDS	KEYS=(2Ø,2),	
7			E		SORT		
8					EX	*MSEQFSØ,	
9					SEQOUT	HMA=22ØØØ,	
10					FILE	WORK, NAME=EXWORK, DEVADD=(Ø4,2),	
11					FILE	OUT, NAME=NXTEDIT,	
12			E		EXITS	OPEN=22ØØ1, ITEM=224Ø7, PROG=OUTOCAØ1,	
13							

EXAMPLE 2

This example illustrates a call for a sort followed by a request to create a file on 1/2-inch tape. The work file is "*SORTWORK" located at pcu 04, drive 0. Input to the sort is on tape. Items are blocked two per record; the tape format is bannered, standard labels with odd parity. The output file is to contain sort items. The format of the output tape is unbannered, standard labels with even parity. Items are blocked two, and a tape mark precedes the trailer. The input tape labels are updated and used as output labels. The user may inspect (and modify) the labels for the output tape through own-coding.

EASYCODER
CODING FORM

PROBLEM ________ PROGRAMMER ________ DATE ________ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					EX	*SORT3PØ,	
2					SORT	TRANS=YES,	
3					FILE	IN, NAME=TRANSACT, DEVTYPE=TAPE, ITEM=125,	
4						REC=251,	
5			E		FIELDS	KEYS=(2Ø,2), EXTR=ITEM,	
6					EX	*MSEQFSØ,	
7					SEQOUT		
8					EXITS	OPEN=2ØØØØ, PROG=OWN-CODE,	
9					FILE	OUT, NAME=TRANSACT, DEVTYPE=TAPE, REC=25Ø,	
10			E			TM=1, BAN=NO, PAR=EVEN,	
11							

Programmer's Preparation Information for Sequential Output File Program

FILE ALLOCATION

When the output is to be on mass storage, the sequential output file must be allocated prior to the execution of the sequential output file program. It must be organized as a sequential file. It is also necessary that compatibility exist between the description of the file and the data that is to be handled by the program. If some incompatibility should exist, the output file will not be created. An example of such incompatibility is the case where the item size of the sequential output file is not the same as that of the source file (assuming that an item sort has been requested).

SPECIFYING WORK FILE

When the sequential output file program is called through an Execute statement, the work file name that must be specified is the same as that of the last work file used by the preceding sort. Furthermore, the device address must be that of the last volume of that work file.

MEMORY REQUIREMENTS FOR SEQUENTIAL OUTPUT FILE PROGRAM

The memory requirements for the execution of the sequential output file program depend upon whether the items come from the sort-item file or the source-item file.

Output File from Sort Items

When the preceding sort application was an item sort (EXTR = ITEM) or the application was such that no item address was appended to the sort item (ITADD = NO), it is assumed that the item of the output file is to have the same format as the sort item. Therefore, the original input items will not be accessed and no allowance need be made for buffers for the source file.

Output File from Source Items

If the preceding sort application was not an item sort and an item address was appended to the sort item, it is assumed that the items that form the output file are to come from a mass-storage-resident source file (the input file to the sort). In this instance, a buffer is required for a source block. In addition, it is desirable to allow as much memory space as possible for an intermediate item-storage area.

Calculation of Memory Requirements for Sequential Output File Program

When calculating the memory requirements for the sequential output file program, two formulas are used: one when the sort item is used, and one when the source item is used:

1. Sort item used for the sequential output file program - the minimum memory requirement = PROG + SUP + OS + OBUF + SIS + TYPE.
2. Source item used for the sequential output file program - the minimum memory requirement = PROG + SUP + OS + OBUF + INB +IL + 20 + TYPE.

The definitions of the terms in the above formulas follow:

PROG	=	7100,
SUP	=	Size of Supervisor + 200 (see <u>Supervisor</u> manual),
OS	=	Sort-item block size (see Appendix A),
OBUF	=	Output block size,
INB	=	Source-block size (input file to the sort),

SIS	= Sort-item size,
IL	= Source-item size (input-item), and
TYPE	= 2000, only if typewriter is used.

Highest Memory Location Available to the Sequential Output File Program

The highest memory location available to the sequential output file program (HMSQ) is not always equal to the value the user specifies for the highest memory location (HMA). If own-coding is not present, then HMSQ = HMA. If own-coding is present and the address values (OPEN and ITEM) are greater than the highest memory address parameter or lie below the base of the sequential output file program, then HMSQ = HMA. (Own-coding can only lie below the base of the sequential output file program if this program has been relocated from its present base value of 200 decimal.) If the address values lie below the user-specified highest memory address parameter, then HMSQ = OPEN-1, if the OPEN value is less than the ITEM value; otherwise, HMSQ = ITEM - 1.

OWN-CODING FOR SEQUENTIAL OUTPUT FILE PROGRAM

There are no own-coding exits from the sequential output file program if the program was called by the inclusion of the output File statement among the job control statements for the sort. When the sequential output file program is requested through an Execute statement, however, the own-coding functions described below may be performed. It is important that all own-coding routines which modify index registers X1, X2, X3, and X4 restore them prior to returning control to the sequential output file program, with the exception of index register X1 when an item is being added.

Open Exit Own-Coding

MASS STORAGE OUTPUT: The sequential output file program branches to the location specified by the value of the open parameter when it has read the item in the file description index (*VOLDESCR*) for the sequential output file. There is the option not to exit to the open own-code program for every volume of a multivolume sequential file.

Index register X1 contains the address of the left (high-order) end of the file description index item. If the accessing of the sequential output file requires specifying the password parameter, the item is not made available until the requirement has been met. Principally, the item is made available for inspection and it may be moved to the own-code program area. No punctuation is available in the area it occupies. If the own-code program sets punctuation in that area, it is responsible for clearing that punctuation before returning control to the sequential output file program.

If open exit own-coding is to be executed for successive volumes of the file, the sequential output file program is reentered at the location defined by the contents of the B-address register when the sequential output file program branched to the own-code routine. Open exit own-coding may be terminated after any volume by branching to a location whose address is formed by adding a branchspace constant to the normal address. A Store Control Register (SCR) instruction to the B-address register should be the first instruction of the open exit own-coding.

TAPE OUTPUT: The sequential output file program branches to the location specified by the value of the OPEN parameter when it has read a header from the output tape and updated characters 41-80 of the header as required (that is, duplicated characters 41-80 of the input tape header or duplicated characters 41-80 of the dummy header). Index register X1 contains the address of the leftmost (high-order) end of the label. The label is made available for inspection and modification.

The own-code program may alter characters 41-80 only. A word mark is present on the left end of the sort area occupied by the label. If the own-code program sets other punctuation in the sort area, the program should clear that punctuation prior to returning control to the sort.

Similarly, the sequential output file program branches to the location specified by the value of the OPEN parameter when an end-of-reel or end-of-file trailer has been produced. Characters 31-80 are updated as the application requires. They may be altered through user own-coding. An item mark is present on the location specified by an open address + 1 to indicate that the label is a trailer. Index register X1 contains the address of the leftmost (high-order) end of the label. Punctuation procedures followed by the own-code program are identical to those used when the label is a header.

If open own-coding is to be continued for successive labels of the file, the sequential output file program is reentered at the location defined by the contents of the B-address register when the own-code routine was entered. Open own-coding may be terminated after any label has been accessed by branching to a location whose address is formed by adding a branchspace constant to the normal address. A Store Control Register (SCR) instruction of the B-address register should be the first instruction of the open own-code routine.

Item Exit Own-Coding

The item exit own-coding options that may be exercised depend upon whether the sequential output file program is being created from source items or sort items.

If item exit own-coding is active, a branch is made to the location whose address is specified by the own-code parameter. Index register X1 contains the address of the left (high-order) end of the current sort item. The only punctuation present in the sort item is a word mark in the high-order character of the item (see Figure 4-19).

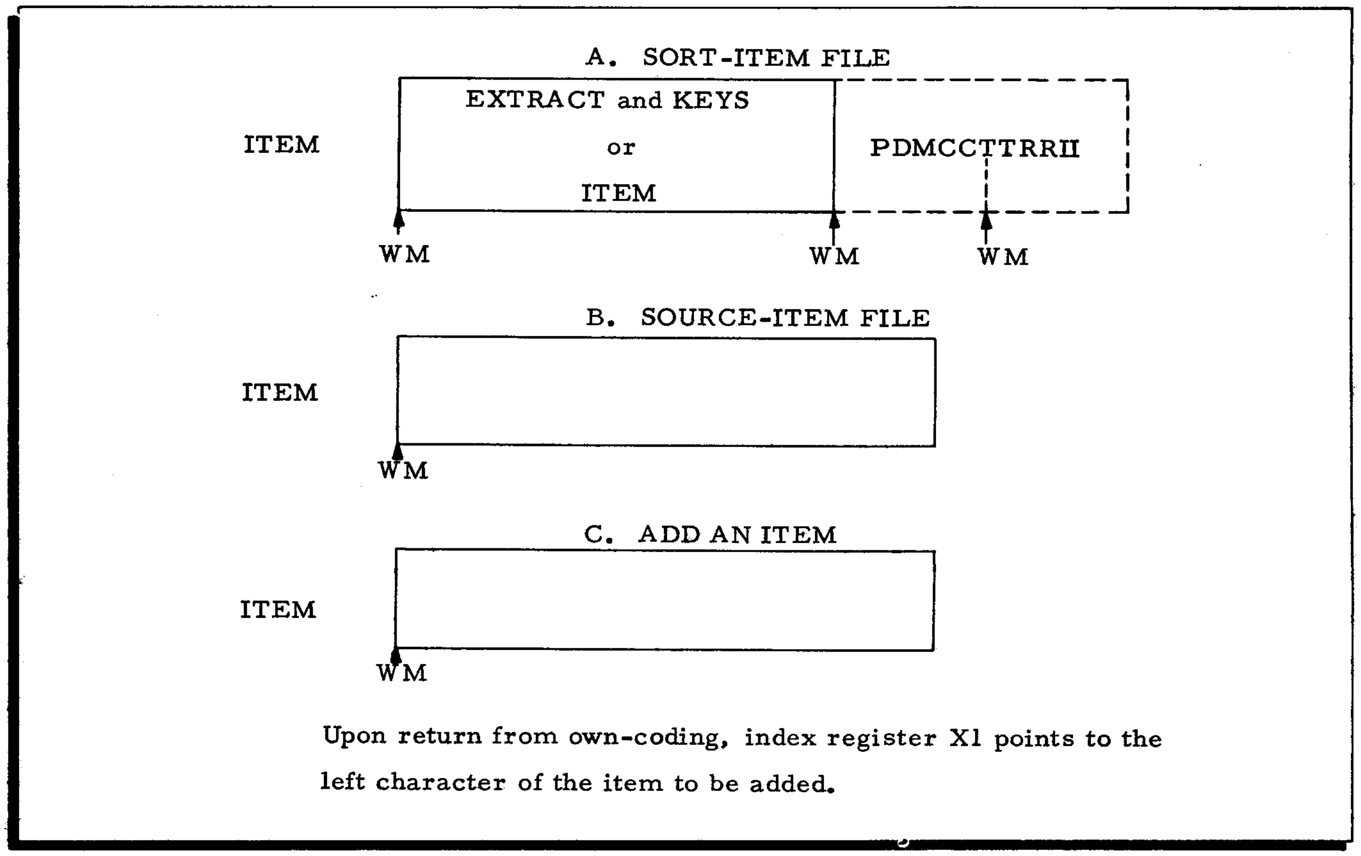


Figure 4-19. Item Punctuation for Sequential Output File Program

The four options in item exit own-coding are listed below and described under separate headings, e.g.:

1. Modifying an item;
2. Adding an item;
3. Deleting an item; and
4. Terminating item exit own-coding.

MODIFYING AN ITEM: The conditions for modifying an item depend upon whether the item used for output is a sort item or a source item, as described below.

1. If a sort item is used for output, the contents of the sort item may be modified, but its length and punctuation cannot be changed. After modification, the user makes a normal return to the sequential output file program.

2. If a source item is used for output, modification of the sort item has no meaning, since the item that is passed to the sequential output file is the source item associated with the current sort item. If the user wishes to modify the original input items, the Fetch macro routine should be used instead of the sequential output file program.

ADDING AN ITEM: The conditions for adding an item depend upon whether the item used for output is a sort item or a source item, as described below.

1. If the sort item is used for output, an item having the same format as the sort-item may be added to the sequential output file program by placing the address of the left (high-order) character of the item in index register X1. A word mark must be present in that high-order character (refer to "Add an Item" in Figure 4-19). The item precedes the current sort item in the output file. Control is returned to the sequential output file program by branching to a location whose address is formed by adding a branch-space constant to the normal return address.
2. If a source item is used for output, an item having the same format as a source item may be added by placing the address of the left (high-order) end of such an item in index register X1. A word mark must be present in the high-order character of the item. The item that is added precedes the source item associated with the current sort item in the sequential output file program. Control is returned to the sequential output file program by branching to a location whose address is formed by adding a branchspace constant to the normal return address.

DELETING AN ITEM: The conditions for deleting an item depend upon whether the item used for output is a sort item or a source item, as described below.

1. If a sort item is used for output, the sequential output file program is re-entered by adding two branchspace constants to the normal return address, provided, that after inspecting the sort item, it is intended to delete that item as a member of the sequential output file program.
2. If a source item is used for output, the sequential output file program is reentered by adding two branchspace constants to the normal return address, provided, that after inspecting a sort item, it is intended that the associated source item should not be included in the sequential output file program.

TERMINATING ITEM EXIT OWN-CODING: When all item own-coding processing has been completed, the own-coding routine must branch to a location whose address is obtained by adding three branchspace constants to the normal return address.

An item mark in the location specified by the own-coding address indicates that the sequential output file program has processed all its input. If, at this point, more items are to be added, the own-code program follows the above procedure specified for adding an item. The sequential output file program continues to exit to the own-code program until the terminated own-coding return or a normal return is made.

OUTPUT TAPE LABELS

Dummy Labels

The paragraphs under "Sequential Output File Program Input/Output Files," in this section, indicate the applications for which dummy labels are created for the output file. Tables 4-7 and 4-8 list the formats for dummy labels.

Table 4-7. Format for Dummy Header

Characters	Contents	Comment
1-5	01 30 24 51 15	1HDRΔ
6-10	ttttt	Tape reel identification: alpha-numeric (preserved from output reel).
11-15	blanks	
16	57 or 40	hyphen: if even parity minus: if odd parity
17-19	sss	Reel sequence number: decimal.
20	blank	
21-30	fffff fffff	File name: 10 alphanumeric characters. In a dummy header, characters 21-30 are blank unless a name is specified through the output file statement.
31-35	yyddd	Creation date obtained from the relevant field of the Supervisor.
36	57 or 40	hyphen: if even parity minus: if odd parity
37-39	ggg	Retention cycle - 0.
40-80		Blanks

Other Labels

The paragraphs under "Sequential Output File Program Input/Output Files" in this section indicate the applications for which input tape labels are updated and used as labels for the output file. The first header of the input file and the last trailer (1EOFΔ) form the basis for the creation of the output file labels. Input tape header characters 1-36 (see comments in Table 4-7 for a description of the fields involved) are updated for the output file header label. The retention cycle for the input tape file is preserved and appears in the output file header labels in characters 37-39. Characters 41-80 are left undisturbed by the sequential output file program, but they may be altered through own-coding.

Table 4-8. Format for Dummy Trailer

Characters	Contents	Comment
1-5	01 25 46 51 15 01 25 46 26 15	1EORΔ 1EORΔ
6-10	rrrrr	Number of data records in this file, decimal. Not accumulated from one reel to the next.
11-20	iiiii iiiii	Number of items processed, decimal. Accumulated for bannered file only.
21-30	fffff fffff	File name: 10 alphanumeric characters. In a dummy trailer, characters 21-30 are blank unless a name is specified through the output file statement.
31-80	blanks	

Input tape trailer characters 1-30 (see comments in Table 4-8 for a description of the fields involved) are updated for the output file trailer labels. Characters 31-80 are undisturbed by the sequential file program, but they can be altered through own-coding.

OPERATING PROCEDURES FOR MASS STORAGE SORT C

Loading and Executing

Job control information can be given to the sort in two ways.

1. The sort parameters are given as statements in the job control file (which must be in the card reader).
2. The sort parameters are resident in memory through some previous user action.

When the sort parameters are given as statements in the job control file, the statements containing the parameters are preceded by an Execute statement. "*SORT3P0" is the required program-segment name to call Mass Storage Sort C. When the parameters are resident in memory through some previous user action, an Execute statement is the only statement required in the job control file. "*SORT3P1" is the program-segment name in this instance.

NOTE: In either of the above cases, the sort can be called through a programmed call to the Supervisor. In this case, no Execute statement is required.

Loading the Fetch Macro Routine

The Fetch macro routine is incorporated into the user's program. The loading of the user's program (including the Fetch macro routine) is performed in the normal manner for the operating system being used. There are no job control statements for the Fetch macro routine. The Fetch routine and its parameters are discussed in detail in this section under "Language Elements of Fetch Macro Routine."

Loading Sequential Output File Program

The sequential output file program can be loaded either directly by a preceding sort or separately by itself.

LOADING SEQUENTIAL OUTPUT FILE PROGRAM BY SORT

When the sort loads the sequential output file program automatically, loading procedures are the same as those for the sort, as previously described under "Loading and Executing." Parameters may be supplied either through job control statements or through main memory.

LOADING SEQUENTIAL OUTPUT FILE PROGRAM SEPARATELY

When the sequential output file program is loaded separately, parameters are entered through job control statements. The job control file for the sequential output file program consists of the following:

1. Execute statement for program segment *MSEQFS0; and
2. Job control statements for the particular application.

Loading and Specializing Sort Phases

Mass Storage Sort C is composed of three logical phases. The sort can be divided into three separate runs if the work files available to the sort are not disturbed.

Delayed Execution of Sort

Each of these three sort runs continues from the point at which the preceding run was terminated. All volumes of the work files must be mounted at the same addresses as for the preceding run.

DELAYED EXECUTION OF MERGE 1-CYLINDER PHASE

If the user wishes to delay running the 1-cylinder phase, the Execute statement, which initiated Mass Storage Sort C by calling for "SORT3," he should include the value "*MERG3M" for the HALT keyword parameter. When segment M0 is loaded, it will not be executed. The values of decimal locations 201 through 209 should be saved. Their values should be reentered prior to the delayed execution of segment M0. The Execute statement to call for the 1-cylinder phase of the sort must have the program-segment name "*MERG3M0."

DELAYED EXECUTION OF MERGE MULTICYLINDER PHASE

This procedure is the same as that described above for the 1-cylinder phase, except that the HALT keyword parameter value is "*MERG3M3," and the Execute statement for the delayed execution must have the program-segment name "*MERG3M3."

Delayed Execution of Fetch Macro Routine

The execution of the Fetch macro routine may be delayed as long as the mass-storage-resident source-item file (the input file to the sort), as well as the work files, are mounted properly. The operation of the Fetch routine in this case is the same as that described in this section under "Loading the Fetch Macro Routine." (If the Fetch routine does not refer to the source file, it need not be mounted during the sort.)

Delayed Execution of Sequential Output File Program

The execution of the sequential output file program may be delayed, provided that the mass-storage-resident source file and the work files are mounted as they were during the sort execution. Loading procedures are the same as described in this section under "Loading Sequential Output File Program Separately." (If the sequential output file program does not refer to the source file, then it need not be mounted during the execution of the program.)

Multivolume Processing

The input file as discussed below is assumed to be mass storage resident.

MULTIVOLUME PROCESSING DURING SORT EXECUTION

During the entire sorting process, the work file must have all volumes online. The source-item file must be online if a key sort is being executed; i.e., the source-item file must have all of its volumes mounted during the sort if there is to be a reaccessing of the source file during a Fetch or sequential output file process. However, if an extract or item sort is being performed, the number of volumes mounted for the source-item file may be the minimum number required for that file. For a sequential or direct access file there may be only one volume mounted, if desired. For indexed sequential files, there may be as few as one additional volume mounted at one time as long as the master/cylinder index and general overflow volume remain mounted throughout the run.

MULTIVOLUME PROCESSING DURING FETCH EXECUTION

During execution of the program containing a specialized version of Fetch, the work files specified for the sort must have all volumes online. Furthermore, those volumes must have the same device addresses as those assigned to them for the sort.

If the application requires access to the original input file, as is necessary in a pure key sort, the volume of the source-item file must be on-line, and each volume must be assigned the same device address that it had during the sort.

MULTIVOLUME PROCESSING DURING SEQUENTIAL OUTPUT FILE EXECUTION

The work files specified to the sort must have all volumes on-line during the execution of the sequential output file program. Work file volumes must also be assigned the same device address as those assigned during sort execution.

When the application of the sequential output file program requires access to the original input file, the source-item file must have all volumes mounted as they were during the sort. The volumes that contain the sequential output file may be mounted one at a time, if desired.

Tape Positioning

INPUT TO THE SORT

The initial reel of a file must be positioned so that the first record to be read is the header of the file with standard labels, or the first data record of the file when no labels are present. This permits the processing of a multifile reel. Successive reels of a multireel file are rewound prior to being opened.

OUTPUT FROM THE SEQUENTIAL OUTPUT FILE PROGRAM

The file is created from the beginning-of-tape, and each reel of the file is rewound prior to opening.

File Protection

The sort requires that the data write switch be set to PERMIT for the control unit containing the work file. The sequential output file program requires that the data write switch be set to PERMIT for the control unit containing the sequential output file. In both cases, it is desirable that the remaining three switches for the control unit be set to PROTECT. The sequential output file must not be a protected file.

Operator Control

Error conditons are indicated by either a halt, defined by the contents of the B-address register if the control panel is being used, or through typed messages if the console is being used. The operator takes corrective action and enters a response character through the control panel if one is being used, or he types the response character at the console keyboard if the typewriter is being used.

The mass storage sort, the Fetch, and the sequential output file program use the mass storage Physical I/O C. The sort and the sequential output file program also use Logical I/O C. Therefore, Physical I/O C and Logical I/O C halts (or messages) may appear in addition to those described in the paragraphs that follow. (Refer to the Data Management Subsystem manual for a description of the halts and messages not listed below.)

CONTROL PANEL ERROR HALTS

The following paragraphs discuss the error halts that are defined by the contents of the B-address register. Such error halts can occur when the control panel is being used.

Input/Output Error Conditions

A B-address register value on the control panel in the range 0000-4777 indicates an input/output error condition.

PERIPHERAL DEVICE CONDITIONS: If the value in the B-address register is in the range 0000 to 3777, the condition relates to a peripheral device.

The general form of the B-address value is ppxd,

pp = peripheral control unit,

x = code indicating type of condition,

- 0 = device not operable,
- 1 = uncorrectable read error,
- 2 = uncorrectable write error,
- 3 = end of storage medium,
- 4 = positioning or addressing error,
- 7 = miscellaneous condition, and

d = device number

These values are consistent with the standard peripheral error halt codes.

The operator should determine the peripheral control XX unit involved and then take appropriate action. If the control unit is mass storage, the halt is executed by the Physical or Logical I/O C of the Mod 1 (MSR) Operating System (refer to the Data Management Subsystem manual). If the control unit is for a device other than mass storage, the B-register values are listed in Table 4-9.

Table 4-9. Peripheral Error Halts for Mass Storage Sort C

B-Address Register	Cause	Action
pp1l	Uncorrectable card read error.	Correct card and reposition it in reader. Enter a G (27_8) into the location specified by the contents of the A-address register. If the card cannot be corrected, enter an E (25_8) into the location specified by the contents of the A-address register. This will initiate a Supervisor search for the next job.
pp1d	Uncorrectable magnetic tape read error.	The contents of the A-address register give the address of a location in memory into which a response character must be entered. Three response characters are valid: G(27_8), A(21_8), or E(25_8). A G response causes the error record to be reread. An A response permits the error record to be bypassed. An E response initiates a search by the Supervisor for the next job. The entering of any other character causes an immediate return to the halt.
pp2d	Uncorrectable write error.	The contents of the A-address register give the address of a location in memory into which the response character must be entered. Two response characters are valid: G(27_8) or E(25_8). A G response causes the record to be rewritten. An E response initiates a search by the Supervisor for the next job. The entering of any other character causes an immediate return to the halt.
pp3d	Halt for multireel input or output.	This halt occurs whenever there is more than one input or output reel and no alternate tape drive specified. If an alternate drive is specified, the program does not halt but waits until the next reel is mounted. Mount the next reel on the drive in use for the file. The only valid response is G; this causes the program to continue.

INPUT/OUTPUT FILE CONDITIONS: If the value of the B-address register is in the range 4000 to 4777, the condition is related to logical operations with files (see Table 4-10).

The general form of the value in the B-address register is 4xyd,

d = device number.

If the value of the B-address register is not listed in Table 4-10, the condition is related to mass storage files. A list of mass storage B-address register values and their meanings can be found in the Data Management Subsystem manual.

Table 4-10. Input/Output File Error Halts

B-Address Register	A-Address List	Cause	Action
406d	Response field - one character. Specific code-one character (72_8). PPd - two characters. Filename-ten characters. Input reel record count-five characters	Input reel record count incorrect.	There are two valid response characters: A(21_8) or E(25_8). An A response allows the program to continue processing. An E response initiates a search for the next program. Any other response causes an immediate return to the halt.
404d	Response field - one character. Specific code - one character (03_8). PPD - two characters. Filename-ten characters.	1HDRΔ not found for input file. (Possible problems are that the input tape is mispositioned or that the wrong tape is mounted.)	There are two valid response characters: G(27_8) and E(25_8). A G response causes a new reel to be reopened but no repositioning is executed if the reel is the initial reel of the input file. An E response initiates a search by the Supervisor for the next job. Any other response causes an immediate return to the halt.
407d	Response field - one character. Specific code-one character (13_8). PPd - two characters. Filename-ten characters. Input reel sequence number - three characters.	Reel sequence incorrect.	There are three valid response characters: A(21_8), E(25_8), or G(27_8). An A response causes the reel to be accepted and allows processing to continue. An E response initiates a search by the Supervisor for the next job. A G response causes a new reel to be reopened, but no repositioning is executed by the program if the reel is the initial reel of the input file. Any other response causes an immediate return to the halt.
405d	Response field - one character. Specific code-one character (71_8). PPd-two characters. Filename-ten characters. yyddd-creation date-five characters.	Retention cycle has not expired.	There are three valid response characters: A(21_8), E(25_8), or G(27_8). An A response causes the mounted reel to be accepted. A G response causes a newly mounted reel to be opened, and if the reel is accepted, processing continues. An E response initiates a search by the Supervisor for the next job. Any other response causes an immediate return to the halt.

Job Control File Error Conditions

A B-address register value on the control panel in the range 5000-5077 indicates error conditions in the job control file that are common to the Mod 1 (MSR) Operating System (see Table 4-11). A B-address register value in the range 5040-5046 indicates that invalid parameters have been fed to the sort. (Any corrective action for the latter assumes that the parameters to the sort were put in the job control file and were not resident in memory when the sort was called.)

Table 4-11. Job Control File Error Conditions

B-Address Register	Cause	Action
5000	Syntactic (grammatical) error, e.g., a right parenthesis is missing.	The contents of the A-address register give the address of a location in memory into which a response character must be entered. Two response characters are valid, $G(27_8)$ or $E(25_8)$. The entering of any other character causes an immediate return to the halt. If the error can be corrected, the set of job control statements should be refed (excluding the Execute statement) and the G response made. When the error cannot be corrected, an E response initiates a search by the Supervisor for the next job.
5001	Statement not recognized.	
5002	Invalid positional parameter.	
5003	Invalid keyword.	
5004	Required parameter missing.	
5005	Invalid parameter value.	
5010	Invalid combination of parameters.	
5040	Illegal parameter value. No further specification given.	The contents of the A-address register give the address of a location in memory into which the response character E must be placed. G or any other response value causes an immediate return to the halt. If the error can be corrected, the set of job control statements preceded by the Execute statement (program-segment name *SORT3P0) should be refed and the E response made. When correction is not possible, the E response causes the Supervisor to search for the next job.
5041	Invalid own-code address. Conflicts with sort memory requirements.	
5042	Insufficient memory space to execute the sort.	
5043	Illegal parameter value for key and extract fields.	
5044	Key and extract fields overlap or are inconsistent with the item size.	
5045	Error in the select and delete parameter values.	
5046	Insufficient work file area.	

Program Specific Error Conditions

A B-address register value in the range 6400-6477 indicates conditions specific to the sort, Fetch macro routine, and the sequential output file program. The corrective action that may be taken varies according to the halt. Therefore, a value in the A-address register gives the address of a list of information. These error conditions are listed in Table 4-12.

Table 4-12. Program Specific Error Conditions

B-Address Register	A-Address List	Cause	Action - A = (21_8), G=(27_8), E=(25_8)
6400	Response field-one character. File name - ten characters.	Incorrect file name or input file not located.	(1) Mass Storage Resident: If an incorrect file name has been given to the sort, the correct name may be entered in file-name and the G response made. Otherwise, an E is the only acceptable response. (2) Tape Resident: Possible problems are that the input is positioned to the wrong file or that the wrong tape is mounted. A G response causes a new reel to be opened, but no repositioning is executed by the program if the reel is the initial reel of the input file. An E response initiates a search by the Supervisor for the next job. Any other response causes an immediate return to the halt.
6401	Response field-one character. Password-eight characters.	Incorrect password.	The valid password may be entered into password and a G response made. Otherwise, an E is the only acceptable response.
6402	Response field-one character. Password-eight characters.	Password required.	The password may be entered into password and a G response made. Otherwise, an E is the only acceptable response.
6403	Response field-one character. File name - ten characters.	Incorrect work-file name or work file cannot be located.	If the error is due to an incorrect work-file name, the correct name may be entered into file-name and a G response given. Otherwise, an E is the only acceptable response.
6404	Response field-one character.	Internal parameter record not found.	A G response will initiate another search for the parameter record. An E response will cause the Supervisor to search for the next job.

Table 4-12 (cont). Program Specific Error Conditions

B-Address Register	A-Address List	Cause	Action - A=(21_8), G=(27_8), E=(25_8)
6405	Response field-one character.	Sequence error.	The only permitted respone is E, thus initiating a search for the next job. The sort must be rerun.
6406	Response field-one character. Error code[1] - one character. Discrepancy[2] - three characters.	Item count discrepancy.	If an A response is given, the sort continues to end-of-job. If an E response is given, control is returned to the Supervisor. The sort output file cannot then be processed through the Fetch routine or the sequential output file program.
6407	Response field-one character. Work-file name - ten characters.	Work file does not have a physical record size of 250 characters or is not a sequential file.	The only permitted response is E, which initiates a search for the next job. The sort must be rerun.
6410	Response field-one character.	Restricted sort-item block size is too small.	If an A response is given, the sort continues using the sort-item block size originally computed. An E response initiates a search for the next job.
6411	Response field-one character. Sort-item block size-four characters (decimal).	A display of sort-item block size.	A G response allows calculation to continue.
6412	Response field-one character. Calculation - ten characters.[3]	A display of results of work area calculation.	If only calculations were requested, then either a G or an E response initiates a search for the next job. However, if the intention was to execute the sort following calculations, the only permitted response is E which initiates a search for the next job.
6414	Response field-one character. File name - ten characters. Device address - three characters.	Work files are assigned to mixed device classes.	A G response causes the file to be reopened. An E response will initiate a search for the next job.
6415	Response field-one character. File name - ten characters. Device address - three characters.	Volume label not found.	A G response initiates a new search for the volume label. An E response initiates a search for the next job.

Table 4-12 (cont). Program Specific Error Conditions

B-Address Register	A-Address List	Cause	Action - A=(21_8), G=(27_8), E=(25_8)
6450	Response field-one character. File name - ten characters.	Incorrect file name or output file cannot be located.	If an incorrect file name has been given to the sort or to the sequential output file program, the correct name may be entered in file-name and the G response made. Otherwise, an E is the only acceptable response.
6451	Response field-one character. Password - eight characters.	Incorrect password for the sequential output file.	The valid password may be entered into password and a G response made. Otherwise, an E is the only acceptable response.
6452	Response field-one character.	Insufficient memory space for sequential execution.	E is the only acceptable response. The sequential output file program must be rerun; but it must either be assigned more memory or the sequential output file must be reallocated with a smaller block size. If the sequential output file program was called through the presence of an Output File statement to the sort, it is not necessary to rerun the sort. Instead, the program should be called through an Execute statement (program-segment-name *MSEQFS0), followed by the appropriate set of job control statements.
6453	Response field-one character.	Item size less than six characters.	The items of the sequential output file do not exceed an item size of six characters. E is the only acceptable response.
6454	Response field-one character.	Parameter records not found.	The only acceptable response is E. A possible cause is that an incorrect file name and device address have been specified for the work-file parameter.
6455	Response field-one character.	The sequential output file has not been allocated correctly.	The only acceptable response is E. An example of the type of error that causes this halt is the case where the item size specified in the file description index for the sequential output file is not equal to the size of the item that is to be placed in that file.

Table 4-12 (cont). Program Specific Error Conditions

B-Address Register	A-Address List	Cause	Action - A=(21_8), G=(27_8), E=(25_8)
6456	Response field-one character. Error code[4] - one character. Discrepancy [5] - three characters.	Item count discrepancy.	This halt occurs if there is an item count discrepancy during the creation of the sequential output file. If an A response is given, the sequential output file program continues to end-of-job.
6457	Response field-one character.	Sequential output file is too small for volume of output.	An A response will close the output file.
6460	Response field-one character. File-name-ten characters.	Work file not located.	This halt occurs if the sequential output file program is being executed and the work file cannot be located. The correct name may be entered into file-name and a G response given. Otherwise an E is the only acceptable response.
6461	Response field-one character.	Invalid own-code address. Conflicts with memory requirements for sequential output file program.	E is the only valid response. The sequential output file program must be rerun. Call the program through an Execute statement (*MSEQFS0), followed by the corrected set of job control statements.
6462	Response field-one character.	Illegal parameter value specified to the sequential output file program.	E is the only valid response. The sequential output file program must be rerun. Call the program through an Execute statement (*MSEQFS0), followed by the corrected set of job control statements.
6471	Response field-one character.	Nonexecutable version of Fetch (see assembly diagnostics).	The only valid response is to enter an E and return control to the Supervisor.
6472	Response field-one character.	Sort item has no appended address, but source processing is requested using Fetch routine.	
6473	Response field-one character.	Buffer sizes too small for actual block size using Fetch routine.	
6474	Response field-one character.	Sort work file cannot be located using Fetch routine.	

Table 4-12 (cont). Program Specific Error Conditions

B-Address Register	A-Address List	Cause	Action - A=(21_8), G=(27_8), E=(25_8)
6475	Response field-one character.	No close parameter was specified but Fetch has completed processing.	The only valid response is to enter an E and return control to the Supervisor.
6476	Response field-one character.	Parameter record not found.	

[1] Error code:
00 = Presort item count exceeds merge item count.
01 = Presort item count is less than merge item count.

[2] Discrepancy - the difference between presort and merge item counts, expressed in binary.

[3] Format of calculation results - RRRRΔDDDDΔ where:
RRRR - in decimal, the total number of 10-track cylinders required for the work area.
DDDD - in decimal, the number of additional 10-track cylinders needed, based upon work area as defined by work-file parameters.

[4] Error code:
00 - Sort item count exceeds sequential output file item count.
01 - Sort item count is less than the sequential output file item count.

[5] Discrepancy - The difference between the two item counts, expressed in binary.

CONSOLE MESSAGES

When a console message indicates an error or requests operator action, the operator performs the following steps.

1. He reads the typeout. (To repeat the message, he presses the space bar twice.) If necessary, he consults the manual for possible action.
2. He performs the desired corrective action.
3. He types the appropriate 1-character response (G, E, etc.).
4. If the typein is correct, he presses the space bar to continue. If incorrect, he types any other character and returns to step 3.

Input/Output Error Conditions

PERIPHERAL DEVICE CONDITIONS: When the first line of the message typed on the console is of the form:

ppΔd message,

then the condition causing the message is related to peripheral device operation.

pp = Address assignment of peripheral control (with the leftmost bit set to zero), and

d = Device number (if applicable).

If pertinent, and if the device is mass storage, a second line of message is typed. This line is a list of information, in the following order.

Specific code	- One character, indicating supplementary information about the condition.
File name	- Ten characters, giving the name of the file being processed.

The operator should determine which peripheral device is referred to by the "pp" and "d" portions of the message. If non-mass storage, Table 4-13 should be consulted to determine appropriate action. If it is mass storage, refer to the Data Management Subsystem manual.

The operator must enter a response character (i.e., type in a character when the TYPE light goes on).

Table 4-13. Peripheral Device Condition Messages

Typewriter Message	Cause	Operator Action
ppΔd READ ERROR	Uncorrectable read error.	Tape: Enter G to reread error record. Enter A to bypass error record. Enter E to initiate a search by the Supervisor for the next job. Cards: Correct card and reposition card in reader. Enter G to reread the card. If the card cannot be corrected, enter an E to initiate a search by the Supervisor for the next job.
ppΔd WRITE ERROR	Uncorrectable write error.	Enter G to rewrite the record. Enter E to initiate a search by the Supervisor for the next job.
ppΔ d END VOLUME	Halt for multireel input or output.	Mount the next reel on the drive in use for the file. The only valid response is G; this causes the program to continue.

INPUT/OUTPUT FILE CONDITIONS: When the first line of message typed out is of the form:

pp d FILE filename, message

then the condition causing the message is related to logical operations with the files.

pp =	Address assignment of peripheral control (with the leftmost bit set to zero);
d =	Device number; and
filename =	Name of the file being processed.

A second line of message is also typed. This line includes the following.

Specific code	- One character, giving supplementary information about the condition.
File name	- Ten characters, giving the name of the file being processed.

The operator should determine which peripheral device is referred to by the "pp" and "d" portions of the message. If non-mass storage, then Table 4-14 should be consulted to determine appropriate action. If it is mass storage, refer to the Data Management Subsystem manual.

The operator must enter a response character.

Table 4-14. Input/Output File Condition Messages

Typewriter Message	Cause	Action
ppΔdΔFILE filename, FILE NOT FOUND Supplementary Information: Specific Code - one character (3). ppd - two characters. File name - ten characters.	1HDRΔ not found for input file. (Possibly, the input tape is mispositioned or the wrong tape is mounted.)	Enter G to reopen a new reel. No repositioning is executed if the reel is the initial reel of the input file. Enter E to initiate a search by the Supervisor for the next job.
ppΔdΔFILE filename, RETENTION CYCLE NOT EXPIRED Supplementary Information: Specific Code - one character (Z). ppd - two characters. File name - ten characters. yyddd - creation date (five characters).	Retention cycle has not expired.	Enter A to accept the mounted reel. Enter G to open a newly mounted reel. If the reel is accepted, processing continues. Enter E to initiate a search by the Supervisor for the next job.
ppΔdΔFILE filename, RECORD COUNT INCORRECT Supplementary Information: Specific Code - one character (@). ppd - two characters. File name - ten characters. Input reel record count-five characters.	Input reel record count incorrect.	Enter A to continue processing. Enter E to initiate a search for the next program.

Table 4-14 (cont). Input/Output File Condition Messages

Typewriter Message	Cause	Action
ppΔdΔFILE filename, VOLUME SEQUENCE NUMBER ERROR Supplementary information: Specific code - one character (=). ppd - two characters. File name - ten characters. Input reel sequence number - three characters.	Reel sequence incorrect.	Enter A to accept the reel and continue processing. Enter E to initiate a search by the Supervisor for the next job. Enter G to reopen a new reel. No repositioning is executed by the program if the reel is the initial reel of the input file.

Job Control File Conditions

When the line typed out is of the form:

JOB CONTROL FILE ERROR, message

then the condition demanding a response is related to the job control file. No second line is typed.

When applicable, the above typeout is followed by:

CANNOT REFEED, Δ

Messages are listed in Table 4-15.

Table 4-15. Job Control File Condition Messages

Typewriter Message	Cause	Action
JOB CONTROL FILE ERROR, SYNTAX	Syntactic (grammatical) error; e.g., a right parenthesis is missing.	If the error can be corrected, refeed the set of job control statements (excluding the Execute statement) and enter G. If the error cannot be corrected, enter E to initiate a search by the Supervisor for the next job.
JOB CONTROL FILE ERROR, COMMAND FIELD	Statement not recognized.	
JOB CONTROL FILE ERROR, POSITIONAL PARAMETER	Invalid positional parameter.	
JOB CONTROL FILE ERROR, KEYWORD PARAMETER	Invalid keyword.	
JOB CONTROL FILE ERROR, MISSING PARAMETER	Required parameter missing.	
JOB CONTROL FILE ERROR, PARAMETER VALUE	Invalid parameter value.	
JOB CONTROL FILE ERROR, PARAMETER COMBINATION	Invalid combination of parameters.	
JOB CONTROL FILE ERROR, PARAMETER VALUE	Illegal parameter value. No further specification given.	If the error can be corrected, refeed the set of job control statements

Table 4-15 (cont). Job Control File Condition Messages

Typewriter Message	Cause	Action
JOB CONTROL FILE ERROR, OWN-CODE ADDRESS	Invalid own-code address. Conflicts with sort memory requirements.	preceded by the Execute statement (program-segment-name *SORT3P0) and enter E. If correction is not possible, enter E to initiate a search by the Supervisor for the next job.
JOB CONTROL FILE ERROR, INSUFFICIENT MEMORY	Insufficient memory space to execute the sort.	
JOB CONTROL FILE ERROR, PARAMETER VALUE	Illegal parameter value for key and extract fields.	
JOB CONTROL FILE ERROR, KEY OR EXTRACT FIELD	Key and extract fields overlap or are inconsistent with the item size.	
JOB CONTROL FILE ERROR, SELECT OR DELETE FIELD	Error in the select and delete parameter values.	
JOB CONTROL FILE ERROR, INSUFFICIENT WORK FILE	Insufficient work file area.	

Conditions Specific to the Sort, Fetch Macro Routine, and Sequential Output File Program

Table 4-16 lists the messages for conditions specific to the sort, the Fetch macro routine, and the sequential output file program.

Table 4-16. Messages for Conditions Specific to the Sort, Fetch Macro Routine, and Sequential Output File Program

Typewriter Message	Cause	Action
INPUT FILE NOT FOUND Supplementary information: File-name - ten characters.	Incorrect file name or input file not located.	Mass Storage Resident: A G response will reinitiate the search for the file. An E response returns control to the Supervisor. Tape Resident: Enter G to open a new reel. No repositioning is executed by the program if the reel is the initial reel of the input file. Enter E to initiate a search by the Supervisor for the next job.

Table 4-16 (cont). Messages for Conditions Specific to the Sort, Fetch Macro Routine, and Sequential Output File Program

Typewriter Message	Cause	Action
PASSWORD ERROR Supplementary information: Password - eight characters.	Incorrect password.	A G response will reinitiate the search for the file. An E response returns control to the Supervisor.
PASSWORD ERROR Supplementary information: Password - eight characters.	Password required.	
WORK FILE NOT FOUND Supplementary information: File-name - ten characters.	Incorrect work-file name or work file cannot be located.	
PARAMETER RECORD NOT FOUND	Internal parameter record not found.	Enter G to initiate another search for the parameter record. Enter E to cause the Supervisor to search for the next job.
SEQUENCE ERROR	Sequence error.	The only permitted response is E. The sort must be rerun.
ITEM COUNT DISCREPANCY Supplementary information: Error code[1] - one character. Discrepancy[2] - three characters.	Item count discrepancy.	Enter A for sort to continue to end-of-job. Enter E to return control to Supervisor. The sort output file cannot then be processed through the Fetch routine or the sequential output file program.
WORK FILE ALLOCATION Supplementary information: Work-file-name - ten characters.	Work file does not have a physical record size of 250 characters or is not a sequential file.	The only permitted response is E. The sort must be rerun.
SORT-ITEM BLOCK RESTRICTION	Restricted sort-item block size is too small.	Enter A for the sort to continue using the sort-item block size originally computed. Enter E to initiate a search for the next job.
SORT-ITEM BLOCK SIZE Supplementary information: Sort-item block size - four characters (decimal).	A display of sort-item block size.	Enter G to allow calculation to continue.
CALCULATION RESULTS Supplementary information: Calculation[3] - ten characters.	A display of results of work area calculation.	If calculations only were requested, enter G or E to initiate a search for the next job. If the intention was to execute the sort following calculations, the only acceptable response is E.

Table 4-16 (cont). Messages for Conditions Specific to the Sort, Fetch Macro Routine, and Sequential Output File Program

Typewriter Message	Cause	Action
WORK DEVICES INCOMPATIBLE Supplementary information: File name - ten characters. Device address - three characters.	Work files are assigned to mixed device classes.	A G response causes the file to be reopened. An E response returns control to the Supervisor.
VOL LABEL NOT FOUND Supplementary information: File name - ten characters. Device address - three characters.	Volume label not found.	A G response initiates a new search for the volume label. An E response initiates a search for the next job.
OUTPUT FILE NOT FOUND Supplementary information: File name - ten characters.	Incorrect file name or output file cannot be located.	A G response will reinitiate the search for the file. An E response returns control to the Supervisor.
PASSWORD ERROR, OUTPUT FILE Supplementary information: Password-eight characters.	Incorrect password for the sequential output file.	
INSUFFICIENT MEMORY, SEQUENTIAL OUTPUT PROGRAM	Insufficient memory space for sequential execution.	E is the only acceptable response. The sequential output file program must be rerun; but it must be assigned more memory or the sequential output file must be reallocated with a smaller block size. If the sequential output file program was called through the presence of an Output File statement to the sort, it is not necessary to rerun the sort. Instead, the program should be called through an Execute statement (program-segment-name *MSEQFS0), followed by the appropriate set of job control statements.
ITEM SIZE LESS THAN 6	Item size less than six characters.	Only E is acceptable.
PARAMETER RECORD NOT FOUND	Parameter records not found.	
OUTPUT FILE ALLOCATION	The sequential output file has not been allocated correctly.	

Table 4-16 (cont). Messages for Conditions Specific to the Sort, Fetch Macro Routine, and Sequential Output File Program

Typewriter Message	Cause	Action
ITEM COUNT DISCREPANCY Supplementary information: Error code[4] - one character. Discrepancy[5] - three characters.	Item count discrepancy.	Enter A for sequential output file program to continue to end-of-job.
OUTPUT FILE TOO SMALL	Sequential output file is too small for volume of output.	Enter A to close the output file.
WORK FILE NOT FOUND Supplementary information: File name - ten characters.	Work file not located.	A G response will reinitiate the search for the file. An E response returns control to the Supervisor.
OWN-CODE ADDRESS - SEQUENTIAL OUTPUT PROGRAM	Invalid own-code address. Conflicts with memory requirements for sequential output file program.	Only E is valid. The sequential output file program must be rerun. Call the program through an Execute statement (program-segment-name *MSEQFS0), followed by the corrected set of job control statements.
PARAMETER VALUE, SEQUENTIAL OUTPUT PROGRAM	Illegal parameter value specified to the sequential output file program.	Only E is valid. The sequential output file program must be rerun. Call the program through an Execute statement (program-segment-name *MSEQFS0), followed by the corrected set of job control statements.
FETCH CANNOT BE EXECUTED	Nonexecutable version of Fetch.	Enter E.
FETCH CANNOT ACCESS SOURCE FILE	Sort item has no appended address, but source processing is requested using Fetch routine.	
BUFFER SIZE INCORRECT - FETCH	Buffer sizes too small for actual block size using Fetch routine.	
WORK FILE NOT FOUND	Sort work file cannot be found using Fetch routine.	
FETCH CANNOT CLOSE	No close parameter was specified, but Fetch has completed processing.	

Table 4-16 (cont). Messages for Conditions Specific to the Sort, Fetch Macro Routine, and Sequential Output File Program

Typewriter Message	Cause	Action
PARAMETER RECORD NOT FOUND	Parameter record not found.	Enter E.

[1]Error code:
00 = Presort item count exceeds merge item count.
01 = Presort item count is less than merge item count.

[2]Discrepancy - the difference between presort and merge item counts, expressed in binary.

[3]Format of calculation results - RRRRΔDDDDΔ where:
RRRR - in decimal, the total number of 10-track cylinders required for the work area.
DDDD - in decimal, the number of additional 10-track cylinders needed, based upon work area as defined by work-file parameters.

[4]Error code:
00 - Sort-item count exceeds sequential output file item count.
01 - Sort-item count is less than the sequential output file item count.

[5]Discrepancy - The difference between the two item counts, expressed in binary.

SECTION V

MASS STORAGE EDIT C

Mass Storage Edit C (hereinafter referred to as the edit program) is a utility routine in the Series 200/Operating System - Mod 1 (Mass Storage Resident).

The edit program cannot be run in a multiprogramming environment.

EQUIPMENT REQUIREMENTS FOR MASS STORAGE EDIT C

Basic Equipment Requirements

The following equipment is required for the edit program:

A Series 200 central processor;

Advanced Programming Instructions;

A minimum of 12,288 characters of main memory; and

One disk device and associated control.

This combination may be any one of the following.

Device Type	Control Type
155	157C, 257C
258	257, 257-1, 260
259	257, 257-1, 260
273	257, 257-1, 260
259A	257A
259B	257B
261	260
262	260

Also required are one card reader and one printer.

Additional Usable Equipment

The following additional equipment also may be used:

One Type 220-1, -3 Console (console I/O requires 4,096 additional characters of main memory);

FUNCTIONAL DESCRIPTION OF MASS STORAGE EDIT C

The edit program provides a printed representation of the physical data stored on any mass storage device. Parameters to the edit program specify the device address and the areas to be edited. The parameters are entered through the job control file, and the areas to be edited are bounded below and above by the cylinder and track values of the starting and terminating address parameters. An alphabetic or an octal listing can be specified.

The edit program does not respect the conventions of the Data Management Subsystem. Therefore, a user wishing to edit a specific file should use File Support C (refer to the Data Management Subsystem manual).

Features of Mass Storage Edit C

HEADER LINE

The header line of the edit program contains the following information printed at the top of each page:

1. Peripheral address assignment of the control (pos. 3-12),
2. Device number (pos. 16-25),
3. Pack address (pos. 29-38),
4. Title "MSΔEDIT" (pos. 52-60), and
5. Page number (pos. 99-110).

HEADER LINE RECORD

This line contains the following decimal information:

1. Cylinder number (pos. 14-21),
2. Track number (pos. 25-34),
3. Flag character of the record header (showing protection status), as recorded on disk (pos. 38-43),
4. Record number (pos. 47-60), and
5. Data length (pos. 64-79).

DATA PORTION LINE

This line contains the following information:

1. Read error information (pos. 1-3); appears on first line of record only;
2. Character positions (pos. 7-10); indicates position in record of first character in this line; and
3. Data characters of record (pos. 14-113); this line type is repeated as necessary to exhaust the record.

END-OF-JOB LINE

This line contains the following information:

1. END EDIT (pos. 14-26),
2. Number of tracks processed (pos. 30-50),

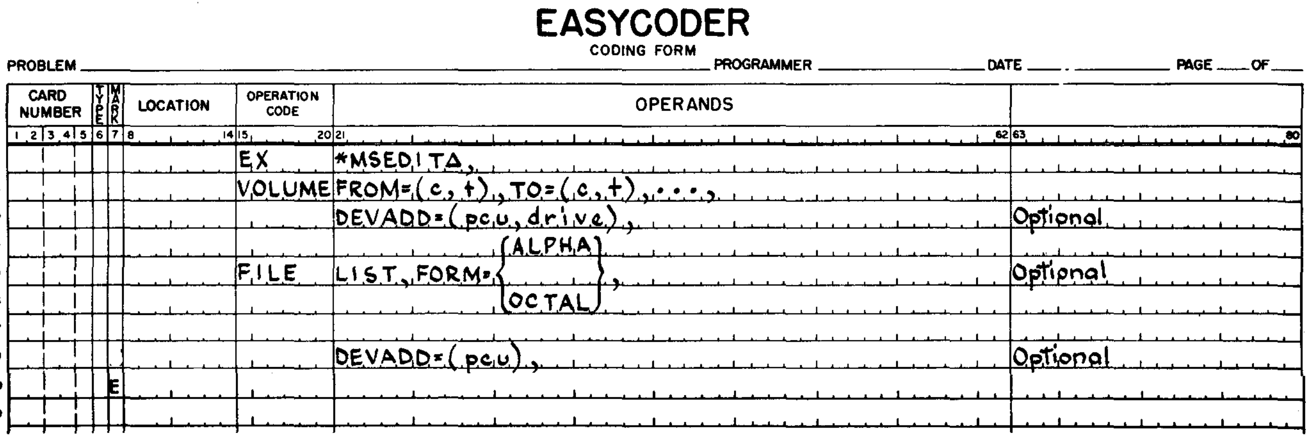
3. Number of records processed (pos. 54-75), and
4. Number of errors (pos. 79-99).

JOB CONTROL LANGUAGE FOR MASS STORAGE EDIT C

Requesting an Edit Operation

The edit program can print as many as six areas from one mass storage device in one operation. In order to edit from more than one device, the edit program must be requested again.

The specification of the parameters to the edit program, i.e., mass storage areas, format of the listing, etc., is made through statements in the job control file (see Figure 5-1). The job control file must be in a card reader.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ___ . ______ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					EX	*MSEDITΔ,	
2					VOLUME	FROM=(c,t),TO=(c,t),....,	
3						DEVADD=(pcu,drive),	Optional
4						ALPHA	
5					FILE	LIST,FORM= { ALPHA / OCTAL },	Optional
6						OCTAL	
7							
8						DEVADD=(pcu),	Optional
9			E				
10							

Figure 5-1. Job Control Statements for Mass Storage Edit C

Two job control statements are required: an Execute statement to load the edit program, and a Volume statement to give the parameters. If the edit is to be in octal form or if the control unit address of the printer is not 02, then the File statement is also required.

EXECUTE STATEMENT

The format of the Execute statement is shown in Figure 5-1.

VOLUME STATEMENT

The Volume statement (see Figure 5-1) gives the parameters of the edit operation. One Volume statement describes a maximum of six areas to be edited, as long as they are all on the same mass storage device. The parameters are described in the following paragraphs. Optional parameters have an assumed or "default" value. When such a parameter is omitted from the job control statements, the default value is used.

The Volume and File statements may appear on one or more lines (for a card deck, one "line" is one card). The last line must have an E in the mark field. The line containing the E may be the same line as the last line containing a parameter or may be a following line containing no parameters.

FROM and TO Parameters

One pair of FROM and TO parameters describes one mass storage area to be edited; there must be at least one such pair. However, one Volume statement may contain as many as six pairs describing the areas to be edited.

The "c" value of the parameter is the cylinder address in decimal; the first cylinder on a device is numbered zero. The "t" value of the parameter is the track address in decimal; the first track on a cylinder is numbered zero. The FROM and TO parameters take the form:

```
FROM = (c,t), TO = (c,t), ...
```

For a given pair of FROM and TO parameters, the area to be edited includes all the tracks and cylinders stated in the parameter (inclusive). All records on each track are edited, including the track-linking record.

The device type, as defined in column 12 of the volume label of the volume being edited, determines the maximum cylinder and track values allowable. The permissible values are shown in Table 5-1. If the values specified in the FROM and TO parameters exceed these values, an illegal parameter halt or a console message occurs.

Table 5-1. Permissible Cylinder-Track Values for Edit

	Volume Label Column 12 Symbol			
Device Type	Octal	Alpha	Range of Cylinder Values	Range of Track Values
155	21	A	0 - 202	0 - 1
258	11	9	0 - 103	0 - 9
259, 259A, 259B	12	'	0 - 202	0 - 9
273	13	=	0 - 202	0 - 19
261	31	I	0 - 127	0 - 127
262	32	;	0 - 127	0 - 127
261L	33	.	0 - 127	0 - 127
262L	34	)	0 - 127	0 - 127

NOTE: Some cylinders which may be edited are reserved for systems use and are not allocatable.

Device-Address Parameter

The device-address parameter (DEVADD) specifies the physical address of the mass storage device being edited. The peripheral address assignment of the control unit (pcu) is written as two octal digits. The I/O bit is not required, but all other bits must be specified, including the sector bits. The drive number of the mass storage device is written as one octal digit. When the device address parameter is omitted, the assumption is that the pcu is 04 and the drive number is zero. The device-address parameter takes the following form.

```
DEVADD = (pcu, drive),
```

FILE STATEMENT

The File statement (see Figure 5-1) has the I/O function name "LIST" as the first parameter of the statement. This statement is not required, unless octal printing or a non-standard print device number is desired.

Form Parameter

The form parameter (FORM) specifies the format of the edit listing. When written as "FORM=OCTAL," an octal listing is produced. When written as "FORM=ALPHA," an alphabetic listing is produced. When the form parameter is omitted, the listing is alphabetic. The form parameter has the following format.

$$\boxed{\text{FORM} = \left\{\begin{matrix}\text{ALPHA}\\\text{OCTAL}\end{matrix}\right\},}$$

Device-Address Parameter

The device-address parameter (DEVADD) specifies the physical address of the printer. The peripheral address assignment of the control unit (pcu) is written as two octal characters. When this parameter is omitted, the pcu value 02 is assumed. The form of the device address parameter is as shown below.

```
DEVADD = (pcu),
```

Examples of Use of Mass Storage Edit C

In the following example, the printer is on pcu 03 and the listing desired is ALPHA. A File statement is necessary. Two areas will be listed. The edit program is resident on a disk device (0), while the volume being edited is on pcu 04, drive 1. The first area listed consists of 12 tracks; i.e., tracks 02 through 05 on cylinders 37 through 39. The second area listed consists of six tracks; i.e., tracks 01 through 06 on cylinder 00.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					EX	*MSEDITΔ,	
2					VOLUME	DEVADD=(Ø4,1),	
3						FROM=(37,2),TO=(39,5),	
4						FROM=(Ø,1),TO=(Ø,6),	
5					FILE	LIST,	
6			E			DEVADD=(Ø3),	
7							

In the following example, the printer is on pcu 02 and the listing desired is ALPHA. No File statement is necessary. The edit program is resident on a tape (drive 0). The volume being edited is on the default device address (04,0). A total of 14 tracks will be listed (tracks 03 through 09 on cylinders 01 and 02).

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1	*MSED	I	T	ΔØΔΔΔΔΔ	ΔΔΔ*		
2			E		VOLUME	FROM=(1,3),TO=(2,9),	
3							
4							

In the following example, *MSEDIT is executed from a resident file on device 0 (boot 64). The volume being edited is on device 2 (second sector) and must be a Type 261 or 262 Disk File. A total of 24 tracks are edited in octal form (tracks 14 through 19 on cylinders 88 through 91).

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1					EX	*MSEDITΔ,	
2					VOLUME	DEVADD=(64,2),	
3						FROM=(88,14),TO=(91,19),	
4			E		FILE	LIST,FORM=OCTAL,	
5							

OPERATING PROCEDURES FOR MASS STORAGE EDIT C

Loading Mass Storage Edit C

The program-segment name for the edit program is "*MSEDITΔ."

LOADING FROM MASS STORAGE

The Execute statement must precede the job control statements for the edit run.

LOADING FROM MAGNETIC TAPE

A console call card (see second example) must precede the job control statements for the edit run when executed under control of Floating Tape Loader-Monitor C.

OPERATOR CONTROL

Error conditions are indicated either by a halt defined through the contents of the B-address register or by a console message.

Control Panel Halts

When a control panel halt occurs and when the appropriate corrective action has been taken, the operator responds by entering a character in the location specified by the A-address register.

Table 5-2 lists halt codes and operator responses for the edit program.

Table 5-2. Halt Codes and Operator Responses for Mass Storage Edit C

B-Address Register	Cause	Action[1]
pp0d	The device is inoperable.	Cycle up the device and enter a G response.
pp1d	Card read error.	Correct the card, reposition it in the card reader, and enter a G to continue. If the card cannot be corrected, enter an E to exit to the Supervisor.
pp2d	Write error on printer.	Enter a G response to attempt to print again. Enter an E response to terminate job.
pp7d	A device malfunction such that edit processing cannot continue.	The only acceptable response is E, which initiates a search by the Supervisor for the next job.
5000	Syntactic (grammatical) error in a job control statement.	Correct the statement causing the error and refeed the control statements. Enter a G into the response location. (The Execute statement must not be included.)
5001	Invalid statement.	Entering an E into the response location initiates a search by the Supervisor for the next job.
5003	Invalid keyword.	
5040	Illegal parameter value.	Cards cannot be refed. Enter a G into the response location to continue processing parameters. Entering an E into the response location initiates a search by the Supervisor for the next job.

Table 5-2 (cont). Halt Codes and Operator Responses for Mass Storage Edit C

B-Address Register	Cause	Action[1]
6600	More than six pairs of FROM and TO parameters specified.	The halt occurs for each pair exceeding six. Entering a G response causes the halt to be repeated until no more pairs are encountered. The edit then processes the first six pairs specified in the Volume statement. Entering an E returns control to the Supervisor.
[1]G = (27_8), E = (25_8).		

Console Messages

When a console message indicates an error or requests operator action, the operator performs the following steps.

1. He reads the typeout. (To repeat the message, he presses the space bar twice.) If necessary, he consults the manual for possible action.
2. He performs the desired corrective action.
3. He types the appropriate 1-character response (G, E, etc.).
4. If the typein is correct, he presses the space bar to continue. If incorrect, he types any other character and returns to step 3.

Table 5-3 lists console messages and operator responses for the edit program.

Table 5-3. Console Messages for Mass Storage Edit C

Message	Cause	Action
pp d INOPERABLE	The device is inoperable.	Cycle up the device and enter a G response.
pp d READ ERROR	Card read error.	Correct the card, reposition it in the card reader, and enter a G to continue. If the card cannot be corrected, enter an E to exit to the Supervisor.
pp d WRITE ERROR	Error in printing.	Enter a G response to attempt to print again. Enter an E response to terminate job.
pp d MISCELLANEOUS	A device malfunction such that edit processing cannot continue.	The only acceptable response is E, which initiates a search by the Supervisor for next job.
JOB CONTROL FILE ERROR, SYNTAX	Syntactic (grammatical) error in a job control statement.	Correct the statement causing the error and refeed the job control statements. Enter a G. (The Execute statement must not be included.)

Table 5-3 (cont). Console Messages for Mass Storage Edit C

Message	Cause	Action
JOB CONTROL FILE ERROR, COMMAND FIELD	Invalid statement.	Entering an E initiates a search by the Supervisor for the next job.
JOB CONTROL FILE ERROR, KEYWORD	Invalid keyword.	Entering an E initiates a search by the Supervisor for the next job.
JOB CONTROL FILE ERROR, ILLEGAL PARAMETER	Illegal parameter value.	Cards cannot be refed. Enter a G to continue processing parameters. Entering an E initiates a search by the Supervisor for the next job.
TOO MANY AREAS SPECI-FIED	More than six pairs of FROM and TO param-eters specified.	The message occurs twice for each pair exceeding six. Enter-ing a G response causes the message to be repeated until no more pairs are encountered. The edit then processes the first six pairs specified in the Volume statement. Entering an E returns control to the Supervisor.

SECTION VI
DISK/TAPE COPY C

Disk/Tape Copy C is a utility routine of the Mod 1 (MSR) Operating System. It offers a fast method of achieving backup security and reproducing previously generated system volumes.

Disk/Tape Copy C is intended as a faster but physically constrained supplement to the more flexible methods of File Support C. For those situations where logical reorganization of data is desired, the user should consult the File Support C section of the Mod 1 (MSR) Data Management Subsystem manual.

Disk/Tape Copy C copies rectangular areas[1] or named files from a mass storage volume onto either magnetic tape or another mass storage volume. The copy on magnetic tape can be copied back, in whole or in part, to any mass storage volume mounted on a device of the same class as the original input device.

When the copy-by-rectangle method is used, as many as 16 rectangles can be copied. When the copy-named-files method is used, any number of named files can be copied.

To be acceptable for copying, a mass storage volume must have been created according to data management conventions. It must have a track format that identifies the first record on each track as record zero; all subsequent records are consecutively numbered in binary.

Use of Disk/Tape Copy C assumes familiarity with the conventions of the Data Management Subsystem.

EQUIPMENT REQUIREMENTS FOR DISK/TAPE COPY C

Basic Equipment Requirements

The following basic equipment is required:

A Series 200 central processor;

Advanced Programming Instructions;

12,288 characters of main memory (Type 261 and 262 Disk Files require 8,192 additional characters of main memory); and

One disk device and associated control.

[1]A rectangular area is specified by a pair of FROM and TO parameters as described later in this section under "Job Control Language for Disk/Tape Copy C."

This combination may be any one of the following.

Device Type	Control Type
155	157C, 257C
258	257, 257-1, 260
259	257, 257-1, 260
273	257, 257-1, 260
259A[1]	257A
259B[1]	257B
261	260
262	260

[1]When either an input or an output mass storage volume is mounted on a Type 259A or 259B Disk Pack Drive, the user should insert the value 259 in the appropriate device-type parameter. This requirement is noted in this section under "Job Control Language for Disk/Tape Copy C."

Also required are one card reader and one Type 222 Printer.

In addition, at least one of the following is required:

One more disk device, as described above, or one Type 204B Magnetic Tape Unit and one tape control unit.

Additional Usable Equipment

The following additional equipment may be used.

* One Type 220-1, -3 Console (console I/O required 4,096 additional characters of main memory);

Second I/O Sector (Feature 1115 for Model 2200);

Additional disk device(s) and control(s); and

Additional Type 204B Magnetic Tape Units.

FUNCTIONAL DESCRIPTION OF DISK/TAPE COPY C

Disk/Tape Copy C offers three copy options — disk-to-tape copy, tape-to-disk copy, and disk-to-disk copy — and two copy methods — copy-by-rectangle and copy-named-files. Either copy method can be used with any of the copy options, but only one copy options and one copy method can be used during one execution of the routine (e.g., a disk-to-tape copy run using the copy-by-rectangle method or a tape-to-disk copy run using the copy-named-files method).

Copy Options of Disk/Tape Copy C

The three copy options of Disk/Tape Copy C are illustrated in the following paragraphs. A detailed description of all job control language appears under "Job Control Language for Disk/Tape Copy C," later in this section.

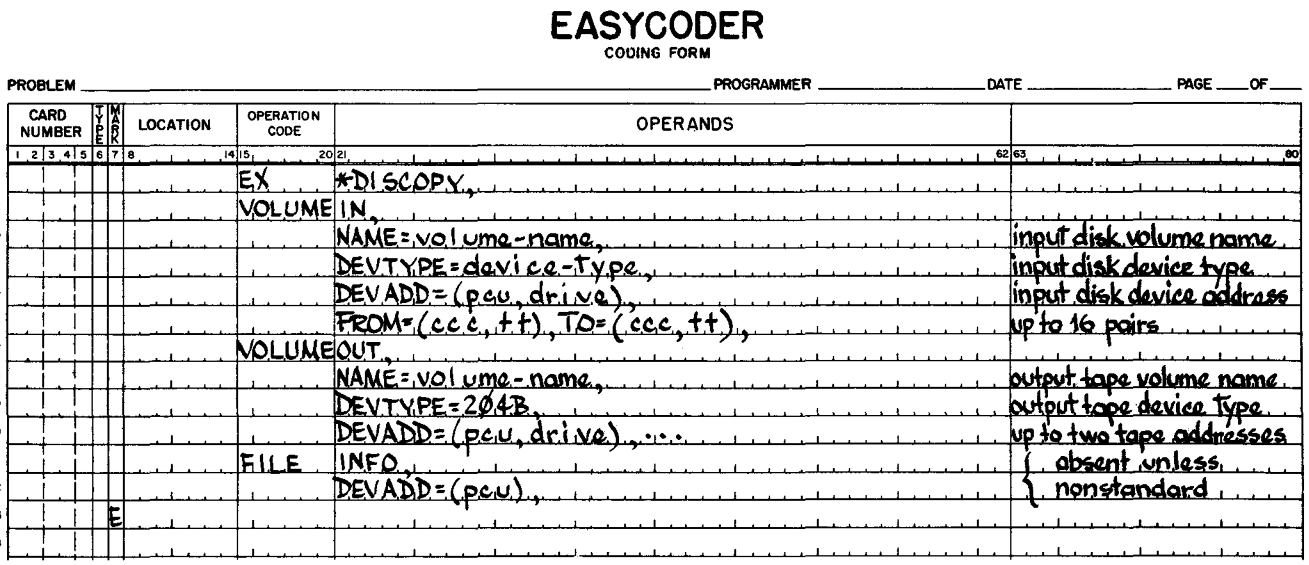
OPTION 1 — DISK INPUT, TAPE OUTPUT (DISK-TO-TAPE COPY)

The disk-to-tape copy option must include either the copy-by-rectangle method or the copy-named-files method (but not both during one execution of the routine). The copy-by-rectangle method is specified by including from one to 16 pairs of FROM and TO parameters in the Input Volume statement. The copy-named-files method is specified by including one File statement for every file to be copied. (Any number of File statements can be included: they must appear between the Input Volume and Output Volume statements.)

If the copy-by-rectangle method is specified, an entire mass storage volume (one rectangle or as many as 16 selected areas (16 rectangles)) can be copied to an output magnetic tape. If the copy-named-files method is selected, any number of named files can be copied from a mass storage volume to an output magnetic tape.

A mass storage volume of a Type 259 Disk Pack Drive requires about 1400 feet of output tape (using a density of 556 BPI); a volume of a Type 261 Disk File requires a 2400-foot reel of output tape for each 13 cylinders copied.

A set of job control statements for a disk-to-tape copy run is shown in Figure 6-1. This figure illustrates the copy-by-rectangle method since a pair of FROM and TO parameters is included. If the copy-named-files method is desired, the pair of FROM and TO parameters is not used; instead, a File statement is used for each file to be copied from disk to tape.



EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					EX	*DISCOPY,	
2					VOLUME	IN,	
3						NAME=volume-name,	input disk volume name
4						DEVTYPE=device-type,	input disk device type
5						DEVADD=(pcu, drive),	input disk device address
6						FROM=(ccc, tt), TO=(ccc, tt),	up to 16 pairs
7					VOLUME	OUT,	
8						NAME=volume-name,	output tape volume name
9						DEVTYPE=204B,	output tape device type
10						DEVADD=(pcu, drive), ...	up to two tape addresses
11					FILE	INFO,	{ absent unless
12						DEVADD=(pcu),	{ nonstandard
13			E				
14							

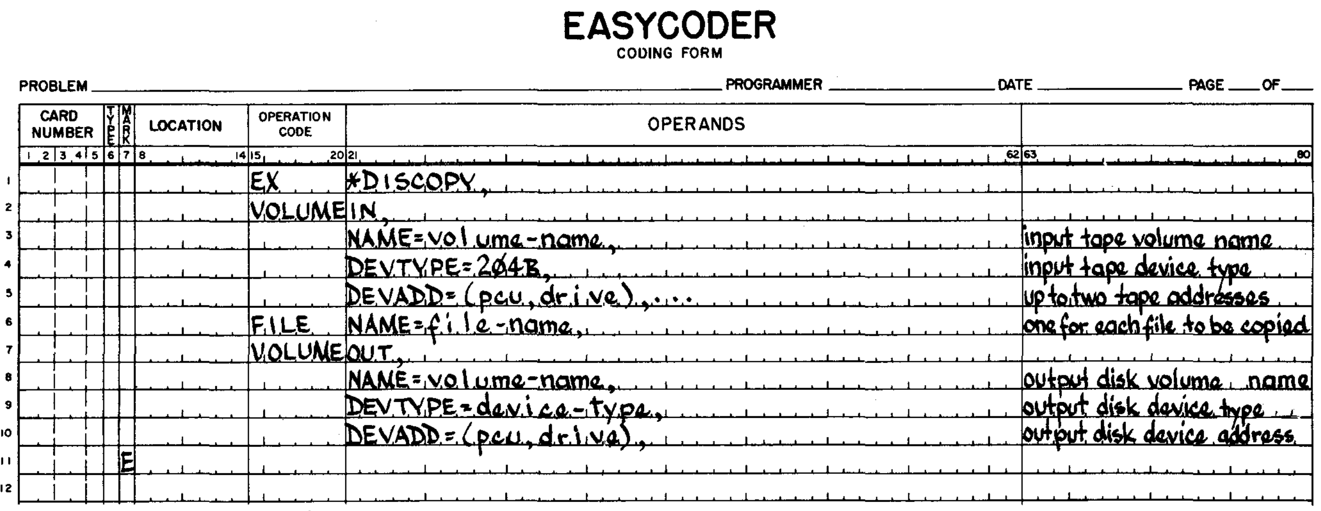
Figure 6-1. Job Control Statements for Disk-to-Tape Copy Option

OPTION 2 - TAPE INPUT, DISK OUTPUT (TAPE-TO-DISK COPY)

The tape-to-disk copy option requires that the input tape be the product of a previous disk-to-tape copy run. The tape-to-disk copy run recopies mass storage rectangles or named files to: (1) the original mass storage volume or (2) any other mass storage volume mounted on a device of the same class as the original input device. Each track in a rectangle or named file is copied to the same cylinder and track address that it occupied on the original input mass storage volume.

A set of job control statements for a tape-to-disk copy run is shown in Figure 6-2. This figure illustrates the copy-named-files method since a File statement is included. (Use of the copy-named-files method requires that the input tape for this run be the product of a previous disk-to-tape copy run that also used the copy-named-files method. Of course, the input tape must contain the file named in the File statement.)

If the copy-by-rectangle method is desired, the File statement is not used; instead, one or more pairs of FROM and TO parameters (one pair for each area to be copied) is used in the Input Volume statement.[1]



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					EX	*DISCOPY,	
2					VOLUME	IN,	
3						NAME=volume-name,	input tape volume name
4						DEVTYPE=204B,	input tape device type
5						DEVADD=(pcu,drive),...	up to two tape addresses
6					FILE	NAME=file-name,	one for each file to be copied
7					VOLUME	OUT,	
8						NAME=volume-name,	output disk volume name
9						DEVTYPE=device-type,	output disk device type
10						DEVADD=(pcu,drive),	output disk device address
11			E				
12							

Figure 6-2. Job Control Statements for Tape-to-Disk Copy Option

OPTION 3 - DISK INPUT, DISK OUTPUT (DISK-TO-DISK COPY)

The disk-to-disk copy option can be used for system generation since it can create an operational output volume that duplicates system files on an input volume. The same result can

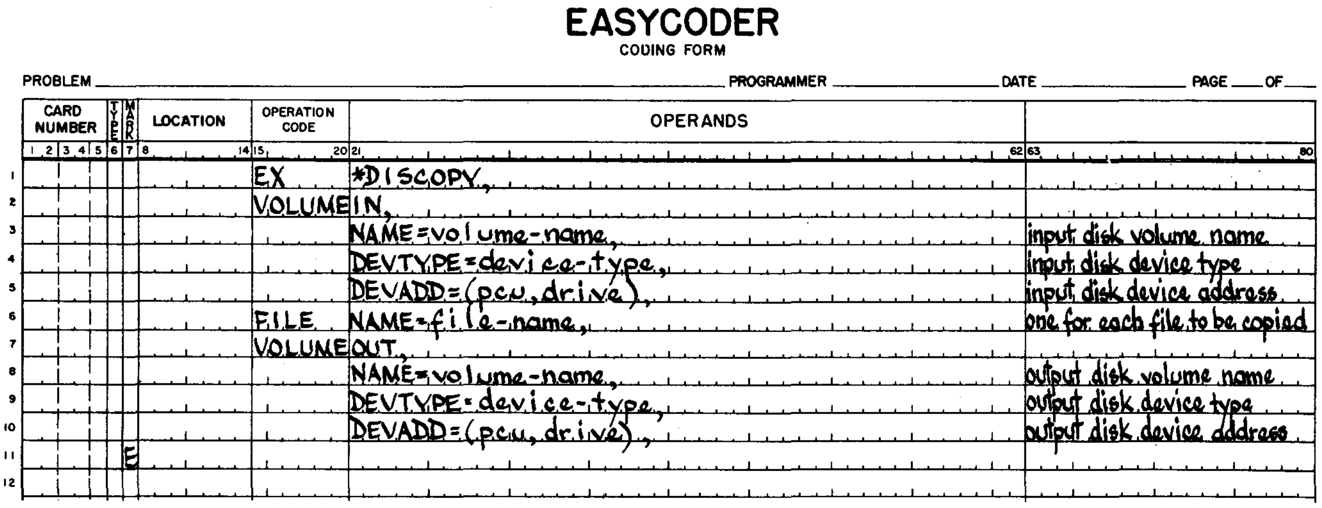
[1]The tape-to-disk copy option does not require that the copy-by-rectangle or copy-named-files method be specified. If neither is specified, the implication is that the same method used to create the input tape will be used during the current tape-to-disk copy run; however, all rectangles or all files originally copied to the tape will now be copied to the designated output mass storage volume.

be achieved, when desired, in a 2-step process by using option 1 (disk-to-tape copy) followed by option 2 (tape-to-disk copy).

For Type 155 Disk Pack Drives the only permissible disk-to-disk copy is from one Type 155 to another. If a user wishes to transfer data between any other disk pack drive or disk file and a Type 155 Disk Pack Drive, the 2-step process using option 1 followed by option 2 must be used.

The disk-to-disk copy option must include either the copy-by-rectangle or the copy-named-files method. The FROM and TO parameter(s) or File statement(s) are specified as shown above under "Option 1 — Disk Input, Tape Output (Disk-to-Tape Copy)." Each track in a rectangle or named file is copied to the same cylinder and track address that it occupied on the input mass storage volume.

A set of job control statements for a disk-to-disk copy run is shown in Figure 6-3. This figure illustrates the copy-named-files method since a File statement is included.



EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE ___ OF ___

	CARD NUMBER 1-5	TYPE 6	MARK 7	LOCATION 8-14	OPERATION CODE 15-20	OPERANDS 21-62	63-80
1					EX	*DISCOPY,	
2					VOLUME	IN,	
3						NAME=volume-name,	input disk volume name
4						DEVTYPE=device-type,	input disk device type
5						DEVADD=(pcu,drive),	input disk device address
6					FILE	NAME=file-name,	one for each file to be copied
7					VOLUME	OUT,	
8						NAME=volume-name,	output disk volume name
9						DEVTYPE=device-type,	output disk device type
10						DEVADD=(pcu,drive),	output disk device address
11			E				
12							

Figure 6-3. Job Control Statements for Disk-to-Disk Copy Option

Copy Methods of Disk/Tape Copy C

Either the copy-by-rectangle method or the copy-named-files method can be used with any of the three copy options, as described below. Only one copy method and one copy option can be used during one execution of Disk/Tape Copy C.

COPY-BY-RECTANGLE METHOD

The copy-by-rectangle method permits the user to specify up to 16 rectangular areas that are to be copied by one of the three copy options. The user defines each rectangular area to be copied by including a pair of FROM and TO parameters in the Input Volume statement of the job control file.

During a tape-to-disk copy run, each rectangle specified to be copied must be equal to or smaller than (i.e., within) a rectangle or unit of allocation that has previously been copied to the input tape. Only one pair of FROM and TO parameters can be specified for each unit of allocation that is to be copied from the input tape.

COPY-NAMED-FILES METHOD

The copy-named-files method permits the user to specify any number of named files that are to be copied by one of the three copy options. The user specifies each file to be copied by including a separate File statement in the job control file.

If the user specifies the copy-named-files method for a tape-to-disk copy run, he must insure that the input tape is the product of a previous disk-to-tape copy run that also used the copy-named-files method.

Precautions When Using Disk/Tape Copy C

The user of Disk/Tape Copy C is responsible for maintaining a logical relationship between a volume directory and the files to which it refers. Specifically, the user must insure that the output mass storage volume directory accurately reflects all rectangles or named files that are copied to that volume.

The volume directory is fully described in Appendix A of the Mod 1 (MSR) Data Management Subsystem manual. The volume label (cylinder 00, track 01, record 0 on the Type 155 Disk Pack Drive; cylinder 00, track 01 on all other mass storage devices) and the volume directory (cylinder 00, track 01, records 1 through 14, and cylinder 01, tracks 00 and 01 on the Type 155 Disk Pack Drive; starts at cylinder 00, track 02 on all other mass storage devices) may be reviewed frequently by means of the map function of File Support C and Mass Storage Edit C.

The following paragraphs describe precautions that must be taken under the copy-by-rectangle and copy-named-files methods when the output volume is mass storage.

COPY-BY-RECTANGLE METHOD

In Disk/Tape Copy C, rectangles copied to an output mass storage volume must be logically consistent with the output volume directory. The output volume directory must contain complete information for the file(s) to which the copied rectangle(s) belong(s).

There are two conditions under which the output volume directory properly reflects the file(s) to which the copied rectangle(s) belong(s).

1. When the file(s) has been previously allocated and loaded onto the current output volume (and not deallocated). The file(s) must have been allocated to the output volume exactly as allocated on the original input volume.
2. When the input volume directory has been copied by rectangle to the output volume.

If the output volume directory does not describe the file(s) to which the copied rectangle(s) belong(s), the data cannot be processed after it is copied. It is not possible to allocate a file to the output mass storage volume after the data rectangles have been copied, since the data would be destroyed.

If a copy-by-rectangle specifies only those tracks constituting the input volume directory, the data area(s) belonging to the file(s) in that volume directory are not automatically copied. To copy the data area(s) of file(s) described in the input volume directory, the user must include the appropriate FROM and TO parameters in the Input Volume statement (or, in a separate execution of Disk/Tape Copy C, the user may include a File statement for each file described in the input volume directory).

COPY-NAMED-FILES METHOD

When the copy-named-files method is used, the file named in each File statement must already be allocated to the output mass storage volume with a structure identical with the file of the same name on the original input mass storage volume. When copied to the output mass storage volume, the named file occupies the same cylinder and track addresses occupied by the same file on the original input mass storage volume.

A file cannot be copied to an output mass storage volume unless it is represented in the output volume directory exactly as it was allocated to the original input volume. If the volume directories have dissimilar entries for the named file, a halt or console message occurs.[1]

JOB CONTROL LANGUAGE FOR DISK/TAPE COPY C

In order to run Disk/Tape Copy C, a job control file consisting of a series of job control statements must be placed in the card reader. Each job control statement contains one or more parameters.

In the following description of job control statements, the term "line" is the equivalent of one card in the job control file.

Sequence of Job Control Statements

Job control statements for Disk/Tape Copy C must appear in the following order:

1. Execute statement (if the routine is loaded from mass storage),
2. Input Volume statement (if present),

[1]Halts and console messages are described in "Operating Procedures for Disk/Tape Copy C," later in this section.

3. File statement(s) (if present),
4. Output Volume statement (if present), and
5. Information File statement (if present).

The Execute statement is required for every run of Disk/Tape Copy C when the routine is loaded from mass storage. All the other job control statements are optional, but each execution of the routine requires either the Input Volume statement or the Output Volume statement.

The last line of the last job control statement must contain an E in its mark field or be followed immediately by a line with blanks in its operation code and operands fields and an E in its mark field.

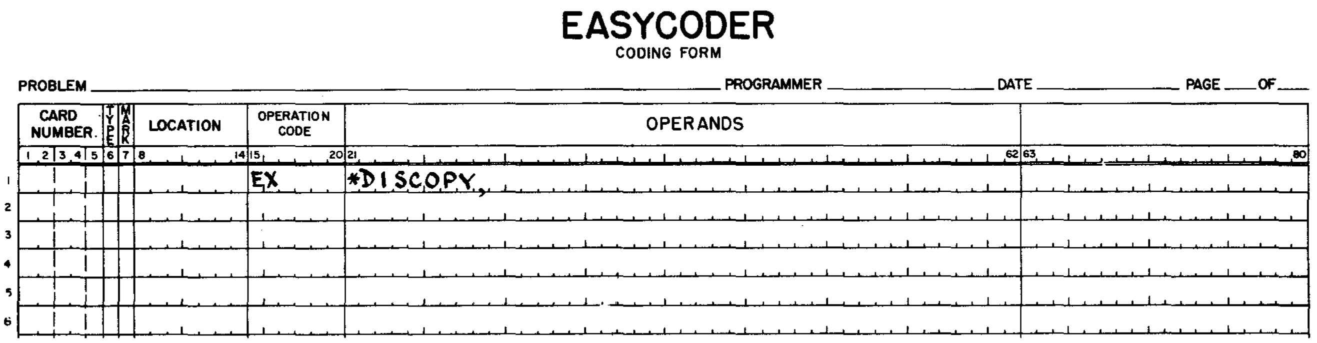
A detailed description of the job control statements follows. Four sample job control files, as they appear when written on Easycoder coding forms, also follow.

<u>Execute Statement</u>

The Execute statement is used only when Disk/Tape Copy C is loaded from mass storage. It directs the Supervisor to load Disk/Tape Copy C from the system disk. An Execute statement is required for every run of this routine when it is mass storage resident.

An Execute statement is not used when Disk/Tape Copy C is loaded from magnetic tape. (For a description of loading from magnetic tape, see "Operating Procedures for Disk/Tape Copy C," later in this section.)

The Execute statement is written as shown in Figure 6-4.



EASYCODER
CODING FORM

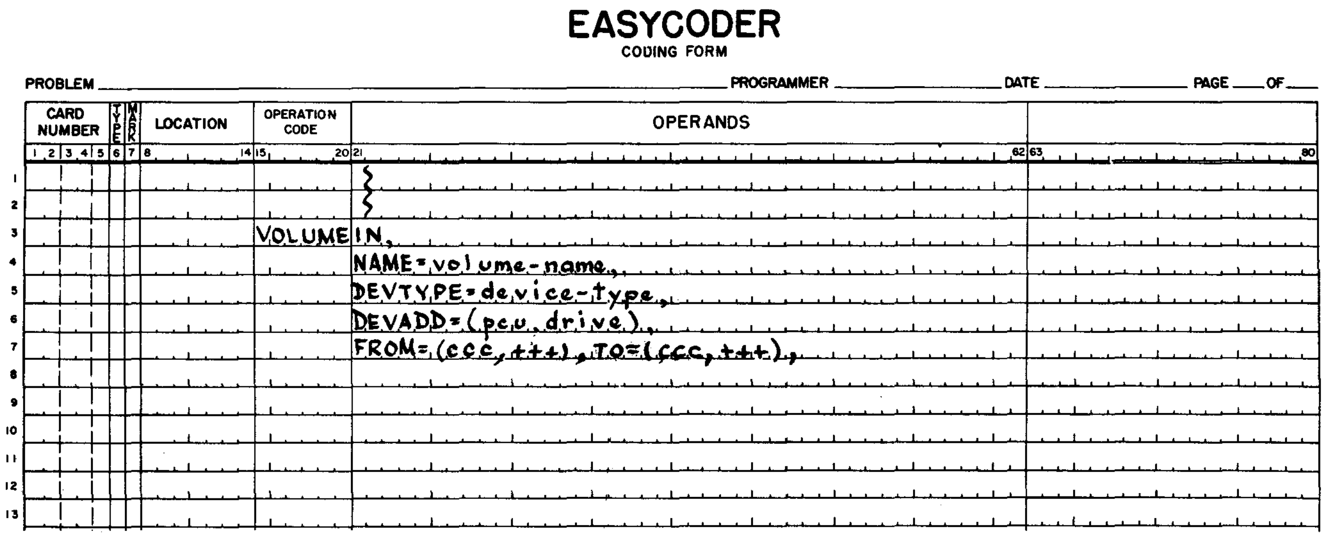
PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER 1 2 3 4 5	TYPE 6	MARK 7	LOCATION 8–14	OPERATION CODE 15–20	OPERANDS 21–62	63–80
1					EX	*DISCOPY,	
2							
3							
4							
5							
6							

Figure 6-4. Execute Statement for Disk/Tape Copy C

<u>Input Volume Statement</u>

A complete Input Volume statement is shown in Figure 6-5.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1						}	
2						}	
3					VOLUME	IN,	
4						NAME=volume-name,	
5						DEVTYPE=device-type,	
6						DEVADD=(pcu,drive),	
7						FROM=(ccc,ttt),TO=(ccc,ttt),	
8							
9							
10							
11							
12							
13							

Figure 6-5. Input Volume Statement for Disk/Tape Copy C

The Input Volume statement is normally present in the job control file since the user usually will desire to specify a value for one or more of the statement's parameters. However, the Input Volume statement is not required.

If the Input Volume statement is omitted, the default values for all its parameters are assumed. In addition, the Output Volume statement must be present if the Input Volume statement is omitted, and the output device must not be a Type 155 Disk Pack Drive.

VOLUME-NAME PARAMETER

The volume-name parameter (NAME) is optional.

When the input is on mass storage, the volume-name parameter refers to the input volume name, and that name is checked before the copy is executed.

When the input is on magnetic tape, the implication is that the tape is the product of a previous disk-to-tape copy run. At that time the name of the input mass storage volume was stored in character positions 52 through 57 of the tape header label. This name is now checked for equality with the name parameter.

Use of this parameter insures that the correct input volume (disk or tape) is present. The form of this parameter is as follows.

$$\boxed{\text{NAME = volume-name,}}$$

volume-name = This name may be up to six characters in length.

The default assumption is that no name checking is required.

DEVICE-TYPE PARAMETER

The form of this parameter is as follows.

$\boxed{\text{DEVTYPE = device-type,}}$

device-type = The type number of the device on which the input volume is mounted.

The device type may be chosen from the following.

Device Type	Medium
155	Mass Storage
258	Mass Storage
259[1]	Mass Storage
273	Mass Storage
261	Mass Storage
262	Mass Storage
204B	Magnetic Tape

[1]When the input volume is mounted on a Type 259A or 259B Disk Pack Drive, the value 259 must be used in the device-type parameter. Omitting the device-type parameter will have the same result, since 259 is the default assumption.

When the value of the input device-type parameter is 155, the only permissible output device-type parameters are 155 or 204B.

If the value of the input device-type parameter is 204B, the implication is that the tape-to-disk copy function is required.

The default assumption is that the input device is a Type 259 Disk Pack Drive.

DEVICE-ADDRESS PARAMETER

The device-address parameter (DEVADD) specifies the address of the peripheral device on which the input volume is mounted. No more than one device-address parameter may be specified for a mass storage device. When input is a multireel tape file, two tape device addresses (primary and alternate) may be specified to permit automatic progression from one tape to another.

Mass Storage	Tape
DEVADD = (pcu, drive) ,	DEVADD = (pcu, drive) , . . .

pcu = Peripheral control unit number, written as two octal characters. The I/O bit is irrelevant, but all other bits must be specified.

drive = Drive number, written as one octal character.

Default Assumptions:

When input is on a mass storage device, the default assumption is pcu 44, drive 0.

When input is on magnetic tape, the default assumption is pcu 40, drive 1.

FROM AND TO PARAMETERS

FROM and TO parameters are used only with the copy-by-rectangle method. If FROM and TO parameters are included in the Input Volume statement, no File statements can appear in the job control file.

One pair of FROM and TO parameters describes one mass storage area (rectangle) to be copied as follows: (1) from an input mass storage volume onto an output magnetic tape or mass storage volume, or (2) from an input magnetic tape (the product of a previous disk-to-tape copy) onto an output mass storage volume.

An Input Volume statement can contain up to 16 pairs of FROM and TO parameters. FROM and TO parameters must follow all other parameters of the Input Volume statement. The form of this parameter is shown below.

```
FROM = (ccc, ttt), TO = (ccc, ttt),...
```

ccc =	The cylinder address, in decimal, where leading zeros are not required. The first cylinder on the device is numbered zero.
+++ =	The track address, in decimal, where leading zeros are not required. The first track on each cylinder is numbered zero.

One pair of FROM and TO parameters specifies that the following mass storage area is to be copied: all the tracks from the track stated in the FROM parameter to the track stated in the TO parameter (inclusive) on each cylinder from the cylinder stated in the FROM parameter to the cylinder stated in the TO parameter (inclusive).

For example, the pair of FROM and TO parameters shown below specifies an area of tracks 15 through 18 on cylinders 12 through 16, a total of 20 tracks.

```
FROM = (12,15) , TO = (16,18) ,
```

When requesting the copy-by-rectangle method for either a disk-to-tape or a disk-to-disk copy run, the user must include at least one pair of FROM and TO parameters in the Input Volume statement.

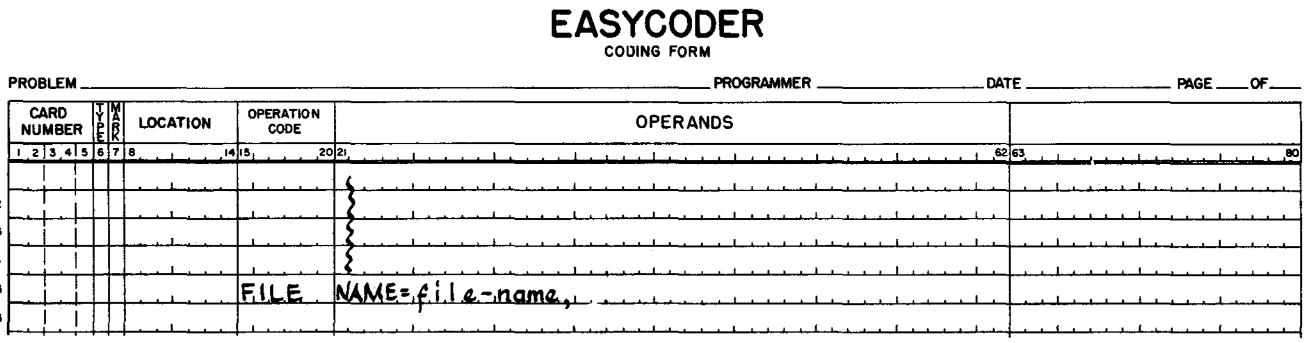
When requesting the copy-by-rectangle method for a tape-to-disk copy run, the user must insure that each pair of FROM and TO parameters defines an area equal to or smaller than (i.e., within) a rectangle or unit of allocation previously copied to the input tape.

NOTES: 1. If the input volume is mass storage and no FROM and TO parameters are specified, one or more File statements must be included in the job control file to implement the copy-named-files method.

2. If the input volume is magnetic tape and neither FROM and TO parameters nor File statements are specified, the same method used to create the tape will be used during the current tape-to-disk copy. However, all rectangles or all files originally copied to the tape will now be copied to the designated output mass storage volume.

File Statement[1]

The File statement is used only with the copy-named-files method. The format of the File statement is shown in Figure 6-6.



EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE ____ OF ____

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1							
2							
3							
4							
5					FILE	NAME=file-name,	
6							

Figure 6-6. File Statement for Disk/Tape Copy C

file-name = The name of the file to be copied (up to ten alphanumeric characters).

When requesting a disk-to-tape copy or a disk-to-disk copy using the copy-named-files method, the user must include a File statement for each file that he desires to copy from the input disk. (No FROM and TO parameters may appear in the Input Volume statement during this execution.) During disk-to-disk copy, the user must insure that each file to be copied has been allocated to the output mass storage volume with a structure identical with that of the corresponding file on the input mass storage volume.

When the user requests a tape-to-disk copy using the copy-named-files method, he must

[1]The following files and tracks cannot be copied by name and must not be named in File statements: (1) the volume directory files, (2) the *BADTRACKS and *VOLSPARES files (which are described in Appendix B of this manual), (3) the volume label track, (4) the bootstrap track, and (5) the substitute bootstrap track.

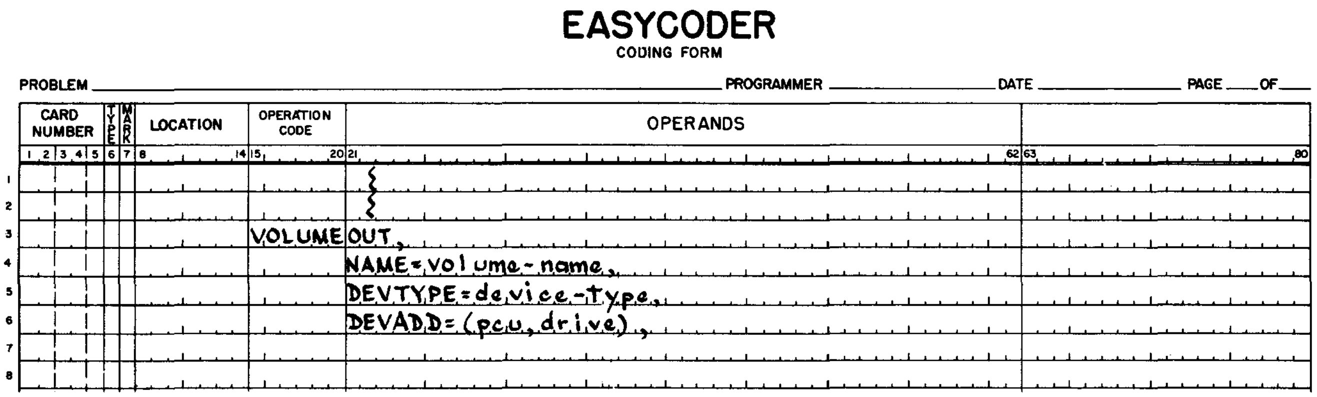
first insure that the input tape is the product of a previous disk-to-tape copy that used the copy-named-files method. A File statement is required for each file that is now to be copied to the output mass storage volume. (No FROM and TO parameters may appear in the Input Volume statement during this execution.) The user must also insure that each file to be copied has been allocated to the output mass storage volume with a file structure identical with that of the corresponding file on the original mass storage volume (i.e., the volume from which the previous disk-to-tape copy was made).

NOTES: 1. If the input volume is mass storage and no File statements are included in the job control file, one or more pairs of FROM and TO parameters must appear in the Input Volume statement to implement the copy-by-rectangle method.

2. If the input volume is magnetic tape and neither File statements nor FROM and TO parameters are included in the job control file, the implication is that the same method used to create the tape will be used during the current tape-to-disk copy run. However, all rectangles or all files originally copied to the tape will now be copied to the designated output mass storage volume.

Output Volume Statement

A complete Output Volume Statement is shown in Figure 6-7.


EASYCODER
CODING FORM

PROBLEM ____ PROGRAMMER ____ DATE ____ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1						⌇	
2						⌇	
3					VOLUME	OUT,	
4						NAME=volume-name,	
5						DEVTYPE=device-type,	
6						DEVADD=(pcu,drive),	
7							
8							

Figure 6-7. Output Volume Statement for Disk/Tape Copy C

The Output Volume Statement is optional unless the Input Volume statement is omitted or the input device-type parameter value is 155. If the Output Volume statement is omitted, the default values for all its parameters are assumed. The Output Volume Statement must be included if the user desires to specify a value for one or more of its parameters.

VOLUME-NAME PARAMETER

The form of this parameter is as follows.

$\boxed{\text{NAME = volume-name,}}$

volume name = This name may be up to six characters in length.

When an output volume is mass storage and an output volume name is specified, this name is checked prior to writing on the output volume; the device type of the output volume is also checked against that specified or defaulted.

When the output volume is magnetic tape and an output volume name is specified, this name is checked prior to writing on the output volume. After the output volume name has been checked, the name of the input mass storage volume is entered into character positions 52 through 57 of the tape header label.

Default Assumptions:

When the output volume is mass storage, the default assumption is that the output volume name and device type are not to be checked.

When the output volume is magnetic tape, the default assumption is that the output volume name is not to be checked. The name of the input mass storage volume is entered into character positions 52 through 57 of the tape header label.

DEVICE-TYPE PARAMETER

The form of this parameter is as follows.

$\boxed{\text{DEVTYPE = device-type,}}$

device-type = The type number of the device on which the output volume is mounted.

The device type may be chosen from the following.

Device Type	Medium
155	Mass Storage
258	Mass Storage
259[1]	Mass Storage
273	Mass Storage
261	Mass Storage
262	Mass Storage
204B	Magnetic Tape

[1]When the output mass storage volume is mounted on a Type 259A or 259B Disk Pack Drive, the value 259 must be used in the device-type parameter.

When the output device is a Type 155 Disk Pack Drive, the only permissible input device-type parameter values are 155 or 204B.

When the output device is a magnetic tape unit, the implication is that the disk-to-tape copy function is required.

Default Assumptions:

If the input volume is mass storage, the default assumption is that the output volume device type is the same as that specified, or defaulted, for the input volume.

If the input volume is magnetic tape, the default assumption is that the output mass storage volume is on a Type 259 Disk Pack Drive.

DEVICE-ADDRESS PARAMETER

The form of this parameter is shown below.

Mass Storage	Tape
DEVADD = (pcu, drive),	DEVADD = (pcu, drive), . . .

pcu = The peripheral control unit number, written as two octal characters. The I/O bit is irrelevant, but all other bits must be specified.

drive = The drive number, written as one octal character.

Only one device-address parameter may be specified for a mass storage device. Two device-address parameters (primary and alternate) may be specified for a magnetic tape device.

Default Assumptions:

When the output volume is mass storage, the default assumption address is pcu 04, drive 0.

NOTE: When performing a disk-to-disk copy, the input and output device addresses must not be identical.

When the output volume is magnetic tape, the default assumption address is pcu 00, drive 1.

Information File Statement

The information file consists of printer messages that occur under certain conditions involving read and write errors.[1] The Information File statement is required only when the printer is assigned to a peripheral control unit whose address is not 02.

If the Information File statement is omitted, the printer is assumed to be assigned to a peripheral control unit whose address is 02.

[1]These messages are described under "Information File Messages," later in this section.

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER 1-5	TYPE 6	MARK 7	LOCATION 8-14	OPERATION CODE 15-20	OPERANDS 21-62	63-80
1						⌇	
2						⌇	
3					FILE	INFO,	
4			E			DEVADD=(pcu),	
5							

Figure 6-8. Information File Statement for Disk/Tape Copy C

DEVICE-ADDRESS PARAMETER

The device-address parameter (DEVADD) must be included when the Information File statement is used. This parameter specifies the address of the printer's peripheral control unit. The form of this parameter is shown below.

$\boxed{\text{DEVADD = (pcu),}}$

pcu = The peripheral control unit number, written as two octal characters. All six bits must be specified.

Sample Job Control Files

The following sample job control files demonstrate four hypothetical copy situations.

EXAMPLE 1 — DISK-TO-DISK COPY

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER 1-5	TYPE 6	MARK 7	LOCATION 8-14	OPERATION CODE 15-20	OPERANDS 21-62	63-80
1					EX	*DISCOPY,	
2					VOLUME	IN,	
3						DEVADD=(Ø4,1),	
4					FILE	NAME=HONEYWELLA,	
5					FILE	NAME=HONEYWELLB,	
6					VOLUME	OUT,	
7						NAME=VOL2,	
8			E			DEVADD=(Ø4,2),	
9							

The action that has been requested is a disk-to-disk copy using the copy-named-files method. Both input and output device types are defaulted to 259. Two files, named HONEYWELLA and HONEYWELLB, are to copied from a volume mounted on pcu 04, drive 1, to a volume mounted on pcu 04, drive 2. Before executing the copy, a check is to be made on the name of the output volume.

EXAMPLE 2A — DISK-TO-TAPE COPY

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
				EX	*DISCOPY,	
				VOLUME	IN,	
					NAME=MASTER,	
				FILE	NAME=BLUBOY1234,	
				VOLUME	OUT,	
					NAME=VOLTPE,	
					DEVTYPE=2Ø4B,	
		E				

The action that has been requested is a disk-to-tape copy using the copy-named-files method. A file named BLUBOY1234 is to be copied from a mass storage volume mounted on a Type 259 Disk Pack Drive (default value) whose address is pcu 44, drive 0 (default value). The output volume is a magnetic tape (since the output device type is specified as a Type 204B Magnetic Tape Unit). The address of the tape unit is pcu 00, drive 1 (default value). Before executing the copy, a check is to be made on the names of the input and output volumes.

EXAMPLE 2B — TAPE-TO-DISK COPY

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
				EX	*DISCOPY,	
				VOLUME	IN,	
					NAME=VOLTPE,	
					DEVTYPE=2Ø4B,	
					FROM=(16,1),TO=(24,9),	
					FROM=(32,2),TO=(38,7),	
				VOLUME	OUT=	
					NAME=NEWMAS,	
		E			DEVADD=(Ø4,1),	

The action that has been requested is a tape-to-disk copy using the copy-by-rectangle method. The input magnetic tape (named VOLTPE) is the product of the disk-to-tape copy run shown in Example 2A. As the result of that copy run, VOLTPE now contains the file named BLUBOY1234. The current example shows how a user can specify that only selected areas of a named file are to be copied to an output mass storage volume. (In this example, the two rectangles specified in the Input Volume statement must be equal to or smaller than (i.e., within) two units of allocation of BLUBOY1234.) The resulting copy transfers only the specified areas (rectangles) of BLUBOY1234 to a mass storage volume named NEWMAS and mounted on a Type 259 Disk Pack Drive (default value) whose address is pcu 04. drive 1.

EXAMPLE 4 — DISK-TO-TAPE COPY

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER 1-5	TYPE 6	MARK 7	LOCATION 8-14	OPERATION CODE 15-20	OPERANDS 21-62	63-80
1					EX	*DISCOPY,	
2					VOLUME	IN,	
3						NAME=DISKA,	
4						DEVTYPE=261,	
5						DEVADD=(Ø4, 2),	
6						FROM=(Ø, Ø), TO=(5, 127),	
7						FROM=(1Ø, Ø), TO=(2Ø, 127),	
8						FROM=(6Ø, 1Ø), TO=(1ØØ, 6Ø),	
9					VOLUME	OUT,	
10						DEVTYPE=2Ø4B,	
11			E			DEVADD=(ØØ, 2), DEVADD=(ØØ, 3),	
12							
13							

The action requested is a disk-to-tape copy using the copy-by-rectangle method. The output is expected to occupy multiple reels.

PROGRAMMER'S PREPARATION INFORMATION

Selecting the Desired Copy Option

Disk/Tape Copy C offers three copy options to the user: disk-to-tape, tape-to-disk, and disk-to-disk. Each of the three copy options permits two copy methods: copy-by-rectangle and copy-named-files.

During a single run of Disk/Tape Copy C, only one copy option and only one copy method can be specified (e.g., the disk-to-tape copy option using the copy-named-files method). The desired copy option and method are specified by the parameters supplied in the job control file, which is placed in the card reader.

Each of the three copy options is a separate segment of Disk/Tape Copy C. After the entire job control file for a run has been processed, the appropriate copy segment is loaded and executed.

DISK-TO-TAPE OPTION

This option is requested by indicating a mass storage device (either specified or defaulted) in the Input Volume statement and by specifying a tape device in the Output Volume statement. Either the copy-by-rectangle or the copy-named-files method must be specified.

The copy-by-rectangle method is specified by including at least one pair of FROM and TO parameters in the Input Volume statement; a maximum of 16 pairs is allowed. The copy-named-files method is specified by including a File statement for every file to be copied; any number of File statements is allowed.

If the copy-by-rectangle method is specified, no File statements may appear in the job control file. If the copy-named-files method is specified, no FROM and TO parameters may appear in the Input Volume statement.

TAPE-TO-DISK OPTION

This option is requested by specifying a tape device in the Input Volume statement and by indicating a mass storage device (either specified or defaulted) in the Output Volume statement.

If the input tape was created by a previous disk-to-tape copy run that used the copy-named-files method, either the copy-by-rectangle or copy-named-files method can be used during the current tape-to-disk copy run. If the copy-by-rectangle method is used, each pair of FROM and TO parameters must be equal to or smaller than (i.e., within) a unit of allocation of the named file. Only one pair of FROM and TO parameters can be specified for each unit of allocation that is to be copied from the input tape.

If the input tape was created by a previous disk-to-tape copy run that used the copy-by-rectangle method, only that method can be used during the current tape-to-disk copy run. If desired, the current copy can include only selected rectangles, parts of selected rectangles, or parts of all rectangles previously copied to the input tape.

If neither the copy-by-rectangle nor copy-named-files method is specified (i.e., the job control file contains neither FROM and TO parameters nor File statements), the input tape will be copied to the output mass storage volume by the same method used in the previous disk-to-tape copy run. However, <u>all</u> rectangles or <u>all</u> files will be copied in their entirety to the output mass storage volume.

DISK-TO-DISK OPTION

This option is requested by indicating a mass storage device (either specified or defaulted) in both the Input and Output Volume statements. Either the copy-by-rectangle or the copy-named-files method <u>must</u> be specified. The requirements and restrictions involved in specifying the copy method are identical with those previously described for the disk-to-tape copy option.

<u>Device Usage</u>

In a single execution of Disk/Tape Copy C, only one device address may be specified for each mass storage volume involved. Thus, a maximum of two mass storage volumes may be specified: one as input and one as output.

When processing tape as input or output, two device addresses (one primary and one alternate) may be specified to facilitate multireel processing. Regardless of whether an alternate tape device address is specified, multireel tape processing can be performed. Multireel tape input requires that the reels be mounted in proper sequence as indicated by their reel sequence numbers.

Magnetic Tape Usage

Disk-to-tape and tape-to-disk copy functions of Disk/Tape Copy C require the use of 1/2-inch magnetic tapes mounted on Type 204B Magnetic Tape Units. When the tape-to-disk copy function is desired, the input tape must be the product of a prior disk-to-tape run of the routine.

Tapes are written in odd parity with bannered records.

TAPE HEADER LABEL

The tape header label is 80 characters in length and is the first record on the tape. Its format is shown in Table 6-1.

Table 6-1. Tape Header Label

Character Positions	Contents
1-5	1 HDR△
6-16	Unused (preserved on output)
17-19	Reel Sequence Number
20	R, F, or Blank[1]
21-30	*DISKCOPY*
31-50	Unused
51	Mass Storage Device Type
52-57	Volume Name
58-80	Reserved for future use

[1]R = the tape is output from a copy-by-rectangle execution.

F = the tape is output from a copy-named-files execution.

Blank = the tape is output from the previous version (copy-by-rectangle) of Disk/Tape Copy C. In this case, all rectangles on the input tape are copied during the tape-to-disk copy run. (No FROM and TO parameters should be included in the Input Volume statement.)

TAPE DATA RECORDS

When an input mass storage volume is copied onto an output tape (regardless of the copy method used), each full track of input information (including header areas and track-linking records) is written as a single record onto the tape. Each tape record has a banner character of 41 (octal).

The format of a sample tape record is shown in Figure 6-9. Assume that the input device is a Type 259 Disk Pack Drive and that an input track contains three 1500-character records and a track-linking record.

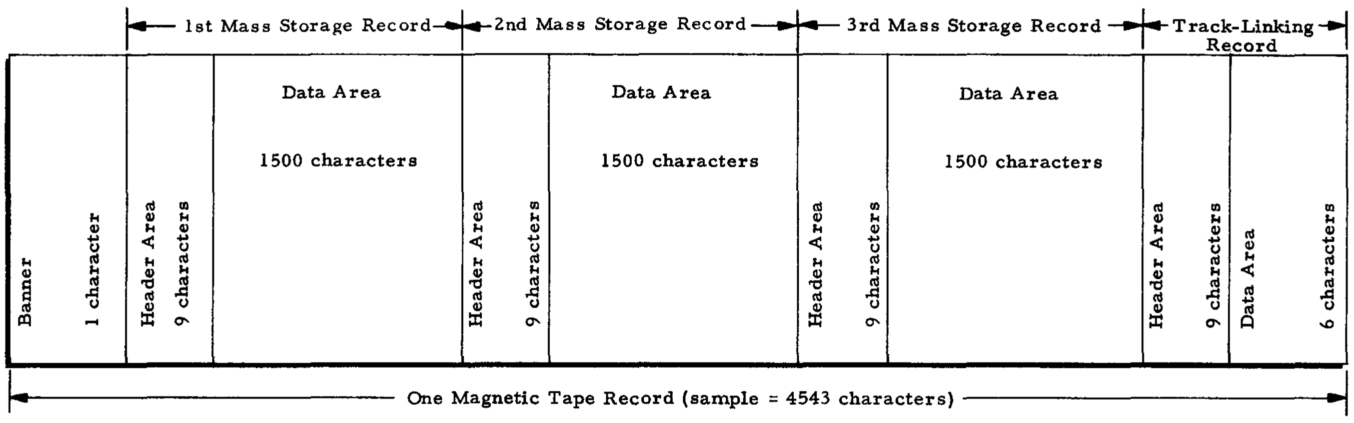


Figure 6-9. Sample Tape Record

TAPE TRAILER LABEL

The trailer label is 80 characters in length and must be the first record after the last data record on the tape. Its format is shown in Table 6-2.

Table 6-2. Tape Trailer Label

Character Positions	Contents
1-5	1EOFΔ or 1EORΔ
6-80	Unused

A 1EORΔ trailer label (followed by two 1ERI records) is written after the last data record on an intermediate reel of output tape for a given copy run.

A 1EOFΔ trailer label (followed by two 1ERI records) is written after the last data record on the last reel of output tape for a given copy run.

TAPE FORMATS

A disk-to-tape copy using either the copy-by-rectangle or the copy-named-files method produces a nonstandard tape format.

Tape Format (Tape Created by Copy-by-Rectangle Method)

The following format appears on the first reel of output tape after a disk-to-tape copy run that used the copy-by-rectangle method.

1. The tape header label
2. An index of the rectangles copied to the output tape(s)
3. A rectangle identification record for one rectangle
4. Data records for this rectangle

   (Items 3 and 4 are repeated for each rectangle copied to this reel)
5. The tape trailer label (followed by two 1ERI records)

If the first reel of output tape is filled before the copy is completed, a second reel of tape is required. The format of the second reel (and any additional reels necessary) is the same as the format of the first reel except that item 2 is not present.

Tape Format (Tape Created by Copy-Named-Files Method)

The following format appears on the first reel of output tape after a disk-to-tape copy run that used the copy-named-files method.

1. The tape header label
2. A file index record for each file copied to the output tape(s)
3. A file identification record for one file
4. A unit of allocation record for this file
5. Data records for this file

   (Items 4 and 5 are repeated for each unit of allocation in the first file copied to this reel. If more than one file is copied to this reel, a file identification record (item 3) appears before the first unit of allocation record (item 4) for each new file)
6. The tape trailer label (followed by two 1ERI records).

If the first reel of output tape is filled before the copy is completed, a second reel of tape is required. The format of the second reel (and any additional reels necessary) is the same as the format of the first reel except that item 2 is not present.

MULTIREEL OUTPUT

Multireel output usually occurs only when copying from Type 261 or 262 Disk Files. An intermediate output reel is terminated by a 1EORΔ trailer label followed by two 1ERI records. The tape is rewound and released.

If an alternate tape drive has been specified, processing automatically continues on the alternate tape.

If no alternate tape drive has been specified, the operator must mount a new tape on the same device address before continuing.

The final output reel is terminated by a 1EOFΔ trailer label followed by two 1ERI records.

Volume Name Processing

Disk/Tape Copy C incorporates the following checks to determine whether the proper input and output volumes are being used.

TAPE INPUT

The header label of an input tape is checked in the sequence outlined below.

1. Character positions 1 through 5 of the tape header label are checked for the presence of 1HDRΔ. If equality results, processing continues to step 2, below. If inequality results, a halt or console message occurs: the user has the option to mount a new tape and try again.
2. Character positions 21 through 30 are checked for the presence of *DISKCOPY*. If inequality results, the tape will not be accepted as input and a halt or console message occurs: the user has the option to mount a new tape and try again. If equality results from the *DISKCOPY* check and an input volume name has been specified, processing continues to step 3. If equality results from the *DISKCOPY* check and no input volume name has been specified, processing continues to step 4 (the input volume name is not checked).
3. Character positions 52 through 57 are checked for equality with the specified input volume name. If equality results, processing continues to step 4. If inequality results, a halt or console message occurs: the user has the option to mount a new tape and try again.
4. Character position 51 is checked for compatibility with the output mass storage device type. If compatibility exists, processing continues to step 5. If compatibility does not exist, a halt or console message occurs: the user has the option to mount a new tape and try again.
5. Character positions 17 through 19 are checked for the proper reel sequence number. If the proper number is present, the tape is acceptable and processing continues. If the proper number is not present, a halt or console message occurs: the user may mount a new tape reel and try again or he may accept the present tape as input to the run.

TAPE OUTPUT

If no name parameter is specified for the output tape, no checking is done on the output tape header label. Instead, a standard Disk/Tape Copy tape header label is created as shown previously in Table 6-1. The name of the input mass storage volume is entered into character positions 52 through 57.

If the name parameter is specified for the output tape, the output tape header label is checked. The contents of character positions 52 through 57 must equal the specified name.

Any invalid contents in the tape header label result in a halt or console message. The user has the option to mount a new output tape and try again or to accept the present output tape.

After the check of the output tape header label is completed, the input mass storage device type is entered into character position 51 and the input volume name is entered into character positions 52 through 57.

MASS STORAGE INPUT

If the name parameter is specified for the input mass storage volume, the volume name is checked. If it is incorrect, a halt or console message occurs: the user may mount a new volume and try again.

MASS STORAGE OUPUT

If the name parameter is specified for the output mass storage volume, the volume name is checked. The resultant action parallels that for mass storage input.

Mass Storage Device Type Check

Whenever a mass storage volume name is checked, the device type (either specified or defaulted) in the same Input or Output Volume statement is also checked against the device type indicated on the volume label.

OPERATING PROCEDURES FOR DISK/TAPE COPY C

Loading Disk/Tape Copy C

Disk/Tape Copy C can be loaded in two ways:

1. From mass storage, via the Supervisor under the Mod 1 (MSR) Operating System;
2. From magnetic tape, via Floating Tape Loader-Monitor C under the Mod 1 (TR) Operating System.

MASS STORAGE RESIDENT

When the routine is loaded from mass storage, an Execute statement (as illustrated in Figure 6-4) must be submitted in the job control file, followed by the desired job control statements for the routine.

TAPE RESIDENT

Loading the routine from magnetic tape is the same as loading from mass storage, except that a console call card is used instead of an Execute statement. The format of the console call card is shown in Table 6-3.

Table 6-3. Console Call Card Format

Characters	Contents	Meaning
1-8	*DISCOPY	Program and segment names of Disk/Tape Copy C.
9	d	Tape drive number from which Disk/Tape Copy C is to be loaded.
18	*	Asterisk (required by the Mod 1 (TR) Operating System).

Protection Switch Settings

Protection switches on mass storage control units must be set as follows:

Disk-to-tape copy — all switches may be set to PROTECT.[1]

Tape-to-disk copy — all switches must be set to PERMIT.

Disk-to-disk copy:

Input volume	— all switches may be set to PROTECT (when separate control units are used for input and output mass storage devices).[1]
Output volume	— all switches must be set to PERMIT.

Processing Bad Areas of a Volume[2]

During various runs of Disk/Tape Copy C, the user may at times encounter (1) uncorrectable read errors on input disk tracks or tape records, or (2) uncorrectable write errors on output disk tracks or tapes. The following two rules describe various possible read and write error

[1]When partitioned sequential files are being copied by the copy-named-files method, protection switches on mass storage control units must be set to PERMIT.

[2]See "Bad Track Handling Procedures for Disk/Tape Copy C" in Appendix B if the *BADTRACKS and *VOLSPARES files are present on a mass storage volume.

conditions. In addition, Table 6-4 provides a useful summary of these conditions.

1. A halt or console message occurs under the circumstances outlined below. The action and printer message that will be produced by an A response depend on which type of error condition is present.

   a. If an input disk track contains data or one or more bad-track TLR's and cannot be successfully read, a halt or console message occurs. If an A response is given, a bad-track TLR is written onto the output volume and a BAD TRACK TLR message appears on the printer.[1] If the unreadable input track occurs at the beginning or in the middle of the area being copied, the bad-track TLR copied to the output volume links to the next consecutive track address within the area. If the unreadable input track is the last track of an area being copied, the bad-track TLR links to the next physical track address (outside the area).

   b. If an input tape record containing data or one or more bad-track TLR's cannot be read successfully, a halt or console message occurs. One can determine the address of the last track (tape record) that was successfully copied onto the output disk by consulting the mass storage cylinder and track address field of the supplementary list (see Table 6-5). If an A response is given, no further action occurs on the output disk for the defective input tape record and a SUBSEQUENT TRACK MISSING message appears on the printer.

      Normally, the missing track occupies the next consecutive physical address following the track listed in the SUBSEQUENT TRACK MISSING message. In some cases, however, the missing track does not occupy the next consecutive physical address. To check the address of the missing track, the user should always refer to the FROM and TO parameters used to create the input tape.

   c. If a disk output write error exists, a halt or console message occurs. If an A response is given, the copied data remains on the faulty output disk track and a FORMAT BAD message appears on the printer.

2. No halt or console message occurs and a BAD TRACK TLR message appears on the printer under the following circumstances:

   a. If an input tape record contains one or more bad-track TLR's and can be read and copied successfully. (The track linkage is the same as that shown in item 1a, above.)

   b. If an input disk track contains one or more bad-track TLR's and can be read and copied successfully. (The track indicated by the bad-track TLR('s) is copied to the output volume. If the output volume is magnetic tape, the header areas of the tape record are modified to show the address of the bad track itself. If the output volume is mass storage, the track indicated by the bad-track TLR('s) is copied to the output disk track whose address corresponds to that of the bad input track.)

[1]The complete format and explanation of each printer message appears in this section under the paragraph, "Information File Messages."

Whenever information is lost because of read or write errors, the cylinder and track address(es) affected should be noted for analysis of the volume's future usability.

The effect of a lost track may be minimal, but, depending on the information it contains, its loss could prevent future runs. For this reason, avoid skipping any track whose utilization is unknown.

Track utilization and contents are determined by the map function ("description" and "unused" options) of File Support C and Mass Storage Edit C.

Table 6-4. Summary of Read/Write Conditions

		Disk Input			Tape Input		
		Read Error	Good Read		Read Error	Good Read	
		Data or Bad-Track TLR('s)	Data	Bad-Track TLR('s)	Data or Bad-Track TLR('s)	Data	Bad-Track TLR('s)
DISK OUTPUT	Good Write	Halt or console message + BTTLR		The track indicated by the bad-track TLR('s) is copied to the output disk track whose address corresponds to that of the original bad track. BTTLR	Halt or console message + STM		BTTLR
	Write Error	Halt or console message + BTTLR	Halt or console message + FB	Halt or console message + FB	Halt or console message + STM	Halt or console message + FB	Halt or console message + FB
TAPE OUTPUT	Good Write	Halt or console message + BTTLR		The track indicated by the bad-track TLR('s) is copied. Header areas of the output tape record are modified to show the address of the bad track itself. BTTLR			
	Write Error	If a tape write error occurs, it is accompanied by a halt or a console message. The user may attempt to rewrite the tape record (G response) or exit to the Supervisor (E response).					

A halt or console message occurs only where indicated. The following abbreviations are used for printer messages:

BTTLR = a BAD TRACK TLR message.
FB = a FORMAT BAD message.
STM = a SUBSEQUENT TRACK MISSING message.

INFORMATION FILE MESSAGES

The information file is a listing on the printer of the BAD TRACK TLR, FORMAT BAD, and SUBSEQUENT TRACK MISSING messages that may occur during an execution of Disk/Tape Copy C. Each message provides a cylinder and track address in the following format:

CYL nnn TR nnn

nnn is a decimal value. The complete format of each message appears below.

1. CYL nnn TR nnn — BAD TRACK TLR
2. CYL nnn TR nnn — FORMAT BAD
3. CYL nnn TR nnn — SUBSEQUENT TRACK MISSING

Operator Communication and Control

When an error condition occurs, 2-way communication between the operating system and the operator is achieved through a control panel or (if certain software and hardware requirements have been met) through a console. If the communication medium is a control panel, an error condition causes the central processor to halt. If the communication medium is a console, an error condition causes a message to appear, but the central processor does not halt.

Halts or console messages are caused by three types of error conditions: peripheral device conditions, job control file conditions, and conditions specific to Disk/Tape Copy C. When a halt or console message occurs, the operator must determine which error condition exists and then make the appropriate response.

CONTROL PANEL HALTS

At the time of a control panel halt, the B-address register contains a value that identifies the error condition. In most cases, the A-address register contains the address of a response location in main memory. The response location is a 1-character main memory location into which the operator must enter a response character appropriate to the error involved. Where indicated, additional information concerning the error can be found in the main memory locations immediately following the response location. This additional information is called the supplementary list (see Table 6-5) and is composed of the following elements (arranged in ascending order of contiguous main memory locations).

Table 6-5. Supplementary List

Field Name	Length
Specific code location	One character (A+1)
Input volume name	Six characters
Input device type[1]	One character
Input pcu address	One character
Input device address	One character
Input magazine address	One character
Output volume name	Six characters
Output device type[1]	One character
Output pcu address	One character
Output device address	One character
Output magazine address	One character
Mass storage cylinder and track address[2]	Six characters in the decimal format cccttt
File name (if applicable)	Ten characters

Table 6-5 (cont). Supplementary List

[1]The device types have the values shown below.
[2]Significant only for mass storage peripheral errors.

Device Type	Value	
	Octal	Alphanumeric
155	21	A
258	11	9
259	12	'
273	13	=
261	31	I
262	32	;
261L	33	.
262L	34	)
204B	63	T

The following paragraphs provide detailed information about the three types of error conditions when the operator's control unit is the control panel.

Peripheral Device Halts

A B-address register value in the range from 0000 through 3777 indicates an error condition involving a peripheral device. The B-address register has the general form ppxd,

pp = Peripheral control unit;

x = Code indicating type of halt condition:

0 = Device not operable,

1 = Uncorrectable read error,

2 = Uncorrectable write error,

3 = End of storage medium,

4 = Positioning or addressing error, and

7 = Miscellaneous condition;

d = Device number.

These values are consistent with the standard peripheral error halt codes.

When a peripheral device error causes a halt, the operator first must determine which peripheral control unit is involved. If "pp" in the B-address register indicates the control unit of a mass storage device, the operator should consult Table 6-6 to learn why the halt has occurred and what action he can take. If "pp" in the B-address register indicates the control unit of a device other than mass storage, the operator should consult Table 6-7.

When the halt condition involves the card reader, the A-address register does not contain meaningful values. In all other cases, the A-address register contains the response location. The supplementary list is also present (see Table 6-5).

Table 6-6. Mass Storage Peripheral Device Halts

B-Address Register Contents	Specific Code (A+1)	Condition	Operator Action[1]
pp0d	01	Device inoperable.	G = retry.
	02	Protection violation.	E = exit to the Supervisor.
pp1d	07	Read error.	G = retry. A = go on to the next track in the current rectangle or unit of allocation. E = exit to the Supervisor.
pp2d	04	Write error.	Same as for pp1d.
pp4d		Positioning or addressing error:	G = retry. E = exit to the Supervisor.
	03	Device error - unable to position to desired cylinder.	
	05	The addressed record has not been located.	
pp6d	06	Record ignored due to misread, or corrupt address mark.	G = retry. E = exit to the Supervisor.
pp7d	11	Miscellaneous device error condition - possible device failure.	G = retry. E = exit to the Supervisor.

[1]The operator must insert the appropriate response character into the response location and press RUN (G = 27_8, E = 25_8, A = 21_8).

Table 6-7. Non Mass Storage Peripheral Device Halts

B-Address Register Contents	A-Address Register Contents	Condition	Operator Action
pp1d	If pp = magnetic tape, response location.	Read error.	1. To reattempt reading, enter G into response location and press RUN. 2. To bypass record and go on to next record, enter A into response location and press RUN.

Table 6-7 (cont). Non Mass Storage Peripheral Device Halts

B-Address Register Contents	A-Address Register Contents	Condition	Operator Action
pp1d (cont)		Read error.	3. To exit to the Supervisor, enter E into response location and press RUN.
	If pp = card reader, 0000.		Correct erroneous card, if possible; refeed cards, starting with that card, and press RUN.
pp2d	If pp = magnetic tape, response location.	Write error.	1. To reattempt writing, enter G into response location and press RUN. 2. To exit to the Supervisor, enter E into response location and press RUN.
	If pp = printer, response location.		An erroneous line has been printed. To ignore error and go on to the next line, enter G into response location and press RUN.
pp3d	pp = magnetic tape, response location.	End-of-reel. (Occurs only when an alternate tape address has not been specified.)	1. To continue, when next reel is ready, enter G into response location, and press RUN. 2. To exit to the Supervisor, enter E into response location and press RUN.
NOTE: G = 27_8, E = 25_8, A = 21_8.			

Job Control File Halts

A B-address register value in the range from 5000 through 5077 indicates an error condition related to the statements of the job control file. The A-address register contains the address of the response location. The operator decides what action is required and enters the appropriate response character into the response location (see Table 6-8). No supplementary list is produced.

Halts Specific to Disk/Tape Copy C

A B-address register value in the range from 6530 through 6547 indicates an error condition that is specific to Disk/Tape Copy C. The A-address register contains the address of the response location. The supplementary list is also present (see Table 6-5).

To determine the reason for the halt and the possible responses he may make, the operator should consult Table 6-9.

Table 6-8. Job Control File Halts

B-Address Register Contents	Condition	Operator Action
5000	Syntactic error.	G = Attempt to perform the operation again. The operator corrects the erroneous card, if possible: refeeds the job control statements, excluding the Execute statement; enters a G into the response location and presses RUN. The program then reprocesses the job control file. E = Exit to the Supervisor.
5001	Invalid command field.	
5002	Invalid positional parameter.	
5003	Invalid keyword.	
5004	Required parameter missing.	
5005	Invalid keyword parameter value.	
5010	Invalid combination or sequence of parameters.	
5040	Same as 5010.	E = Exit to the Supervisor.
NOTE: G = 27_8, E = 25_8.		

Table 6-9. Halts Specific to Disk/Tape Copy C

B-Address Register Contents	Specific Code (A+1)	Condition	Operator Action[1]
6530		Wrong input volume mounted (name check failed).	G = reopen the volume. E = exit to the Supervisor.
6531		Wrong output volume mounted (name check failed).	G = reopen the volume. E = exit to the Supervisor.
6532		Tape reel sequence number is incorrect.	G = reopen the current reel. A = accept the current reel. E = exit to the Supervisor.
6533		Either the input mass storage device type is not as specified (or defaulted) in the job control file, or the device type specified in character position 51 of the input tape header label is incompatible with the output mass storage device.	G = reopen the input volume. E = exit to the Supervisor.

Table 6-9 (cont). Halts Specific to Disk/Tape Copy C

B-Address Register Contents	Specific Code (A+1)	Condition	Operator Action[1]
6534		Output mass storage device is not as specified (or defaulted).	G = reopen the mass storage volume. E = exit to the Supervisor.
6535		The input tape header label is invalid.	G = mount correct tape and retry. E = exit to the Supervisor.

6536		The output mass storage volume is the volume from which the system has been loaded.	G = refeed from the first volume statement.
			A = accept the situation.
			E = exit to the Supervisor.

6537		Insufficient memory is available. Type 155, 258, 259, and 273 Disk Pack Drives require 12,288 characters of main memory. Type 261 and 262 Disk Files require 20,480 characters of main memory.	E = exit to the Supervisor.
6540	01	File name not found on the input mass storage or tape volume.	G = mount correct volume and retry. A = go to the next file. E = exit to the Supervisor.
	02	File name not found on output mass storage volume.	
	03	Conflicting units of allocation or file descriptions.	
	04	File requested to be copied begins on a previous reel (reel mounted out of sequence).	
	05	File being copied does not continue on the current reel (reel mounted out of sequence).	
	06	A discrepancy exists in the file description index of the output mass storage volume during a tape-to-disk copy run. The condition has developed while an attempt is being made to copy all files from the input tape (the tape was created by the copy-named-files method and neither FROM and TO parameters nor File statements were specified in the job control file for the current run).	G = mount correct tape and retry. E = exit to the Supervisor.

Table 6-9 (cont). Halts Specific to Disk/Tape Copy C

B-Address Register Contents	Specific Code (A+1)	Condition	Operator Action[1]
6540 (cont)	07	Error in the volume directory.	G = mount correct volume and retry. A = go to the next file. E = exit to the Supervisor.
	40	No files can be processed. (None of the files designated in File statements has been found on both the input and output volumes.)	G = mount correct volume and retry. E = exit to the Supervisor.
6541	01	This rectangle cannot be processed.	G = mount correct volume and retry. A = go to the next rectangle. E = exit to the Supervisor.
	04	Rectangle requested to be copied begins on a previous reel (reel mounted out of sequence).	
	05	Rectangle being copied does not continue on the current reel (reel mounted out of sequence).	
	40	No rectangles can be processed. (All of the rectangles designated in FROM and TO parameters include tracks outside the rectangles extant on the input tape volume.)	G = mount correct volume and retry. E = exit to the Supervisor.

[1]The operator must insert the appropriate response character into the response location and press RUN (G = 27_8, E = 25_8, A = 21_8).

CONSOLE MESSAGES

The console can be used for operator control when the Supervisor (in the Mod 1 (MSR) Operating System) or Floating Tape Loader-Monitor C (in the Mod 1 (TR) Operating System) has been appropriately specialized and when at least 16,384 characters of main memory are available.[1] If either of these two conditions is not met, the operator's control medium is the control panel.

When the console is used for operator control, an error condition does not cause the system to halt. Instead, diagnostic messages are typed out on the console. The operator takes the following steps.

1. He reads the typeout. (To repeat the message, he presses the space bar twice.) If necessary, he consults the manual for possible action.
2. He performs the desired corrective action.
3. He types the appropriate 1-character response (G, E, etc.).
4. If the typein is correct, he presees the space bar to continue. If it is incorrect, he types any nonspace character and returns to step 3.

The routine continues as directed by the response code. If an invalid response is made, the routine causes the diagnostic message to reappear.

The following paragraphs provide detailed information about the three kinds of messages: peripheral device messages, job control file messages, and messages specific to Disk/Tape Copy C.

Peripheral Device Messages

Console messages specific to peripheral device error conditions are listed in Tables 6-10 and 6-11. The format of these messages is:

pp d MESSAGE TEXT

where: pp = Peripheral control unit,

d = Drive number, and

MESSAGE TEXT specifies the error condition.

A second line of typed information contains the supplementary list described in Table 6-5.

If "pp" indicates the control unit of a mass storage device, the operator should consult Table 6-10. If "pp" indicates the control unit of a device other than mass storage, the operator should consult Table 6-11.

[1] When a Type 261 or 262 Disk File is used, the main memory requirement is always 20,480 characters.

Table 6-10. Mass Storage Peripheral Device Messages

Typewriter Message	Specific Code[1] (alphanumeric)	Condition	Operator Action[2]
pp d INOPERABLE	1	Device inoperable.	G = retry.
	2	Protection violation.	E = exit to the Supervisor.

pp d READ ERROR	7	Read error.	G = retry.
			A = go on to the next track in the current rectangle or unit of allocation.
			E = exit to the Supervisor.

pp d WRITE ERROR	4	Write error.	Same as for read error.

pp d POSITIONING ERROR		Positioning or addressing error:	G = retry.
			E = exit to the Supervisor.

	3	Device error — unable to position to desired cylinder.	
	5	The addressed record has not been located.	

pp d MISREAD ADDRESS MARK	6	This device has not recognized a record header.	G = retry.
			E = exit to the Supervisor.

pp d MISCELLANEOUS	9	Miscellaneous device error condition — possible device failure.	G = retry.
			E = exit to the Supervisor.

[1]The specific code is the first character of the supplementary list.

[2]The operator types the appropriate response character and presses the space bar.

Table 6-11. Non Mass Storage Peripheral Device Messages

Typewriter Messages	Condition	Operator Action
pp d READ ERROR If pp = magnetic tape	Read error.	1. To reattempt reading, type G and press space bar. 2. To bypass record and go on to next record, type A and press space bar. 3. To exit to the Supervisor, type E and press space bar.
If pp = card reader		Correct erroneous card, if possible; refeed cards, starting with that card, and press space bar.

Table 6-11 (cont). Non Mass Storage Peripheral Device Messages

Typewriter Messages	Condition	Operator Action
pp d WRITE ERROR If pp = magnetic tape	Write error.	1. To reattempt writing, type G and press space bar. 2. To exit to the Supervisor, type E and press space bar.
If pp = printer		An erroneous line has been printed. To ignore error and go on to next line, press space bar.
pp d END VOLUME If pp = magnetic tape	End-of-reel. (Occurs only when an alternate tape address has not been specified.)	1. To continue, when next reel is ready, type G and press space bar. 2. To exit to the Supervisor, type E and press space bar.

Job Control File Messages

All messages involving errors in the job control file begin with the words JOB CONTROL FILE ERROR. Table 6-12 contains the complete texts of all job control file messages for Disk/Tape Copy C. No supplementary list is produced.

Table 6-12. Job Control File Messages

Typewriter Message	Condition	Operator Action
JOB CONTROL FILE ERROR, SYNTAX	Syntactic error.	1. Correct the erroneous card, if possible; refeed the job control statements, excluding the Execute statement; type G and press the space bar. 2. If the error cannot be corrected, type E and press the space bar. Control returns to the Supervisor.
JOB CONTROL FILE ERROR, COMMAND FIELD	Invalid command field.	
JOB CONTROL FILE ERROR, POSITIONAL PARAMETER	Invalid positional parameter.	
JOB CONTROL FILE ERROR, KEYWORD PARAMETER	Invalid keyword.	
JOB CONTROL FILE ERROR, MISSING PARAMETER	Required parameter missing.	
JOB CONTROL FILE ERROR, PARAMETER VALUE	Invalid keyword parameter value.	
JOB CONTROL FILE ERROR, PARAMETER COMBINATION	Invalid combination or sequence of parameters.	
JOB CONTROL FILE ERROR, CANNOT REFEED, PARAMETER COMBINATION	Invalid combination or sequence of parameters.	The operator must type E and press the space bar. Control returns to the Supervisor.

Messages Specific to Disk/Tape Copy C

Table 6-13 contains messages for error conditions specific to Disk/Tape Copy C. The typewriter messages in Table 6-13 are followed by a second line of typed information, which contains the supplementary list described in Table 6-5.

Table 6-13. Messages Specific to Disk/Tape Copy C

Typewriter Message	Specific Code (alphanumeric)	Condition	Operator Action[1]

INPUT VOLUME NAME WRONG		Wrong input volume mounted (name check failed).	G = reopen the volume.
			E = exit to the Supervisor.

OUTPUT VOLUME NAME WRONG		Wrong output volume mounted (name check failed).	G = reopen the volume.
			E = exit to the Supervisor.

REEL SEQUENCE CHECK FAILED		Tape reel sequence number is incorrect.	G = reopen the current reel. A = accept the current reel. E = exit to the Supervisor.
WRONG INPUT DEVICE TYPE		Either the input mass storage device type is not as specified (or defaulted) in the job control file, or the device type specified in character position 51 of the input tape header label is incompatible with the output mass storage device.	G = reopen the input volume. E = exit to the Supervisor.
WRONG OUTPUT DEVICE TYPE		Output mass storage device is not as specified (or defaulted).	G = reopen the mass storage volume. E = exit to the Supervisor.
INVALID TAPE FORMAT		The input tape header label is invalid.	G = mount correct tape and retry. E = exit to the Supervisor.

Table 6-13 (cont). Messages Specific to Disk/Tape Copy C

Typewriter Message	Specific Code (alphanumeric)	Condition	Operator Action[1]

SYSTEM RESIDENCE VOLUME IS OUTPUT VOLUME		The output mass storage volume is the volume from which the system has been loaded.	G = refeed from the first volume statement.
			A = accept the situation.
			E = exit to the Supervisor.

INSUFFICIENT MEMORY		Insufficient memory is available. Type 155, 258, 259 and 273 Disk Pack Drives require 16,384 characters of main memory. Type 261 and 262 Disk Files require 20,480 characters of main memory.	E = exit to the Supervisor.
FILE CANNOT BE PROCESSED	1	File name not found on the input mass storage or tape volume.	G = mount the correct volume and retry. A = go to the next file. E = exit to the Supervisor.
	2	File name not found on output mass storage volume.	
	3	Conflicting units of allocation or file descriptions.	
	4	File requested to be copied begins on a previous reel (reel mounted out of sequence).	
	5	File being copied does not continue on the current reel (reel mounted out of sequence).	
	6	A discrepancy exists in the file description index of the output mass storage volume directory during a tape-to-disk copy run. The condition has developed while an attempt is being made to copy all files from the input tape (the tape was created by the copy-named-files method and neither FROM and TO parameters nor File statements were specified in the job control file for the current run).	G = mount correct volume and retry. E = exit to the Supervisor.

Table 6-13 (cont). Messages Specific to Disk/Tape Copy C

Typewriter Message	Specific Code (alphanumeric)	Condition	Operator Action[1]
FILE CANNOT BE PROCESSED (cont)	7	Error in the volume directory.	G = mount correct volume and retry. A = go on to the next file. E = exit to the Supervisor.
	-	No files can be processed. (None of the files designated in File statements has been found on both the input and output volumes.)	G = mount correct volume and retry. E = exit to the Supervisor.
RECTANGLE CANNOT BE PROCESSED	1	This rectangle cannot be processed.	G = mount correct volume and retry. A = go to the next rectangle. E = exit to the Supervisor.
	4	Rectangle requested to be copied begins on a previous reel (reel mounted out of sequence).	
	5	Rectangle being copied does not continue on the current reel (reel mounted out of sequence).	
	-	No rectangles can be processed. (All of the rectangles designated in FROM and TO parameters include tracks outside the rectangles extant on the input tape volume.)	G = mount correct volume and retry. E = exit to the Supervisor.

[1]The operator types the appropriate alphabetic response character and presses the space bar.

SECTION VII
VOLUME MODIFICATION D

Volume Modification D is a utility routine of the Mod 1 (MSR) Operating System. The routine assigns or establishes a substitute track for every user-specified bad track on a mass storage volume. A substitute track is assigned to a bad track that is not part of a file on the mass storage volume. A substitute track is established for a bad track that is part of a file on the mass storage volume.[1]

If the *BADTRACKS and *VOLSPARES files have not already been created on the volume, they must be created by Volume Modification D before any processing of bad tracks can occur.

Just before the end-of-run, the printer lists (1) the address of each bad track recorded in the *BADTRACKS file and (2) the addresses of all substitute tracks.

EQUIPMENT REQUIREMENTS FOR VOLUME MODIFICATION D

Basic Equipment Requirements

The following basic equipment is required:

A Series 200 central processor;

Advanced Programming Instructions;

16,384 characters of main memory;

One disk device and associated control unit.

This combination may be any one of the following.

Device Type	Control Type
155	157C, 257C
258	257, 257-1, 260
259	257, 257-1, 260
273	257, 257-1, 260
259A	257A
259B	257B
261	260
262	260

Also required are one card reader and one printer.

[1] The following actions occur when a substitute track is assigned to, or established for, a bad track:

Assign =	A substitute track is assigned to the bad track from the *VOLSPARES file and an appropriate entry is made in the *BADTRACKS file.
Establish =	The action is the same as that for "assign" plus (1) the substitute track is formatted and (2) the data on the bad track is copied to the substitute track. Each defective data record on the bad track is listed on the printer.

Additional Usable Equipment

One Type 220-1, -3 Console.

FUNCTIONAL DESCRIPTION OF VOLUME MODIFICATION D

Volume Modification D can perform the following functions:

1. Creates the *BADTRACKS and *VOLSPARES files (an option);
2. Checks the surface of all tracks in the *BADTRACKS and *VOLSPARES files (if these files are being created by the current run of Volume Modification D);
3. Assigns or establishes a substitute track for every bad track specified in a Track statement;
4. Performs the following actions <u>if</u> the substitute track is established for a bad track that is part of a file:
    a. Formats the substitute track with the number of records per track and the number of characters per record prescribed for the file,
    b. Transfers <u>all</u> records from the bad track to the substitute track,
    c. Prints out (on the printer) the defective record(s) on the bad track, and
    d. Formats the bad track with at least one bad-track track-linking record (TLR) pointing to the substitute track;
5. Makes entries in the volume directory (if the bad track is in a file) and in the *BADTRACKS file (in all cases) to reflect every use of a substitute track as a replacement for a bad track; and
6. Prints out (on the printer) the address of each bad track recorded in the *BADTRACKS file and the addresses of all substitute tracks.

*BADTRACKS and *VOLSPARES Files

The *BADTRACKS and *VOLSPARES files must be created on the volume when Volume Modification D is run if they have not been previously created by Volume Preparation C or an earlier run of Volume Modification D. The *BADTRACKS and *VOLSPARES files are created by Volume Modification D when the user includes the spares parameter (with an appropriate value) in the job control file.[1]

The *BADTRACKS file contains a permanent list of all substitute tracks and all known bad tracks on the volume. Each item in the *BADTRACKS file contains fields for the following information:

1. The address of the bad track,
2. The address of the track substituted for the bad track,

[1] The spares parameter is described in detail in "Job Control Language for Volume Modification D," later in this section.

3. The file name (if any) of the file that contains the bad track,
4. Status information about the bad track, and
5. The availability status of the substitute track.[1]

The *VOLSPARES file contains the spare tracks to be used as substitutes for bad tracks.

Both the *BADTRACKS file and the *VOLSPARES file are described in the volume directory.

<u>Contingencies for Bad Tracks in Critical Areas</u>

Bad tracks in critical areas of a volume are treated as shown below:

1. If a track within the *BADTRACKS file is bad, a halt or console message occurs.[2] It is not possible to continue the current run of Volume Modification C. The routine must be run again with the RELOCATE option specified in the spares parameter.
2. If any track within the *VOLSPARES file is bad, no halt or console message occurs. The address of the bad substitute track is listed on the printer and that track becomes unavailable for use as a substitute.

<u>Conditions That Prevent Track Substitution</u>

FILE-RELATED CONDITIONS

Under certain conditions, track substitution is not possible when a bad track occurs within a file that is already allocated to the volume. The conditions that prevent the use of substitute tracks are summarized below.

1. <u>All file organizations</u> - Track substitution is not possible for bad tracks within a file that does not contain an integral number of blocks per track.
2. <u>Indexed sequential files</u> - Track substitution is not possible in the following situations.
   a. If the bad track is the last track in the data portion of a cylinder.
   b. If the bad track is the last track in a cylinder overflow area.
   c. If the bad track is the last track in the general overflow area.

A halt or console message occurs when track substitution is requested under any of the conditions above. The file containing the bad track(s) must be reallocated (see the File Support C section of the <u>Mod 1 (MSR) Data Management Subsystem</u> manual).

[1] The substitute track is "available" before it is substituted for a bad track. Thereafter, it is "unavailable."

[2] Control panel halts and console messages are described in "Operating Procedures for Volume Modification C," later in this section.

OTHER CONDITIONS

Track substitution is not possible for a track that is unusable. "Unusable" means that the track's surface is bad and not even one bad-track TLR can be read. It is very unlikely that any track will ever be unusable.[1]

Formatting Substitute Tracks

BAD TRACKS WITHIN AN ALLOCATED FILE

If a bad track lies within a file (e. g., "File A") that is already allocated to the volume, Volume Modification D formats the substitute track with the number of records per track and the number of characters per record prescribed for "File A." The flag character in the header area of each record on the substitute track is set to reflect the file protection assigned to "File A."

The bad track's data is transferred, one record at a time, to the newly formatted substitute track. When a defective record is encountered, its data area is written out on the printer (for the user's information) as well as on the substitute track.

(The volume directory and the *BADTRACKS file are appropriately updated to reflect the establishment of the substitute track.)

BAD TRACKS NOT WITHIN AN ALLOCATED FILE

If a bad track does not lie within an allocated file, the substitute track is not formatted by Volume Modification D.

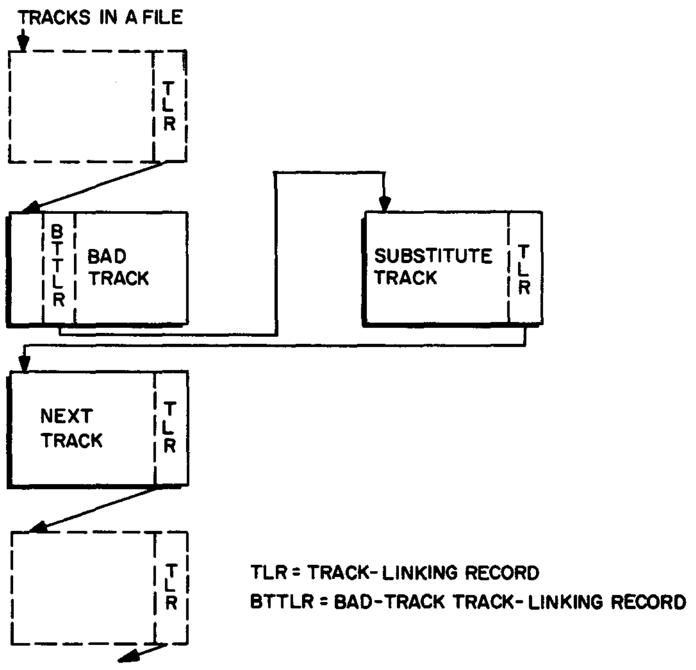
(The *BADTRACKS file is updated to reflect the assignment of the substitute track.)

Linking Substitute Tracks

When a substitute track is established for a bad track in an allocated file, the track-linking record (TLR) on the bad track (pointing to the next track in the file) is written onto the substitute track (causing the substitute track to point to the next track in the file). After the data on the bad track is written onto the substitute track, a bad-track TLR is written onto the bad track, linking it to the substitute track.

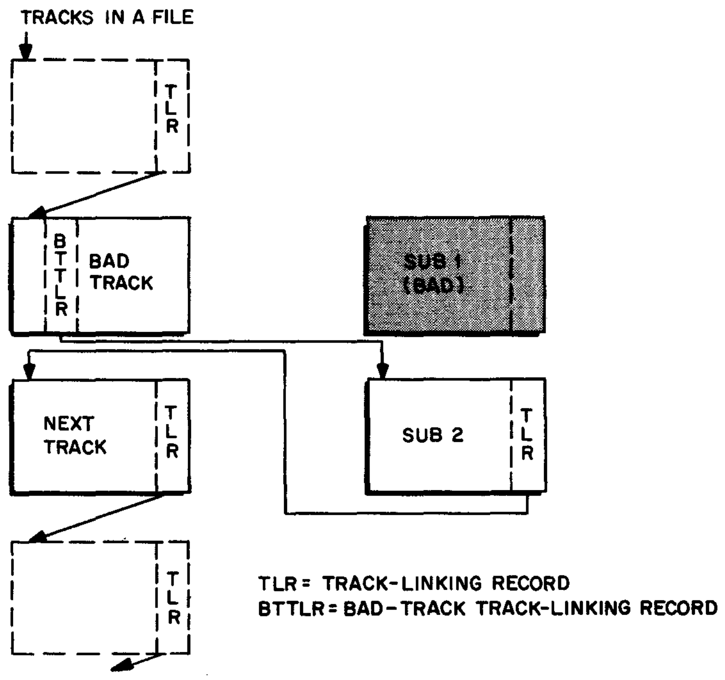
The following diagram shows the resulting track linkage.

[1] If the bootstrap track (cylinder 000, track 000) is unusable, a halt or console message occurs. The volume cannot be used as a system disk because it cannot be bootstrapped; however, the volume can be used for the storage of data. If the volume label track or any track within the volume directory is unusable, a halt or console message occurs. The volume cannot be used for any purpose.



If a substitute track (e.g., Sub 1) subsequently becomes bad and a second substitute (e.g., Sub 2) is established for it, the TLR linking Sub 1 to the next track is written onto Sub 2 (causing Sub 2 to point to the next track). After the data on Sub 1 is written onto Sub 2, a bad-track TLR is written onto the original bad track linking it to Sub 2 (thus, Sub 1 is subsequently bypassed altogether).

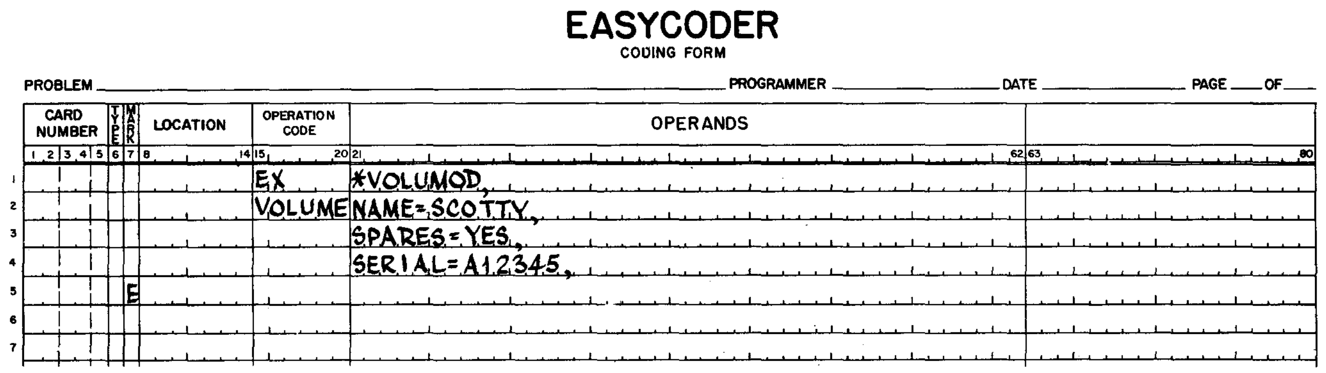
The resulting track linkage is shown below.



Uses of Volume Modification D

The following examples illustrate how Volume Modification D can be run to accomplish four different objectives. In each case, job control language for the routine is shown as it is coded on an Easycoder coding form. Detailed information about job control statements and their parameter values is provided in "Job Control Language for Volume Modification D," later in this section.

EXAMPLE 1 - CREATING *BADTRACKS AND *VOLSPARES FILES



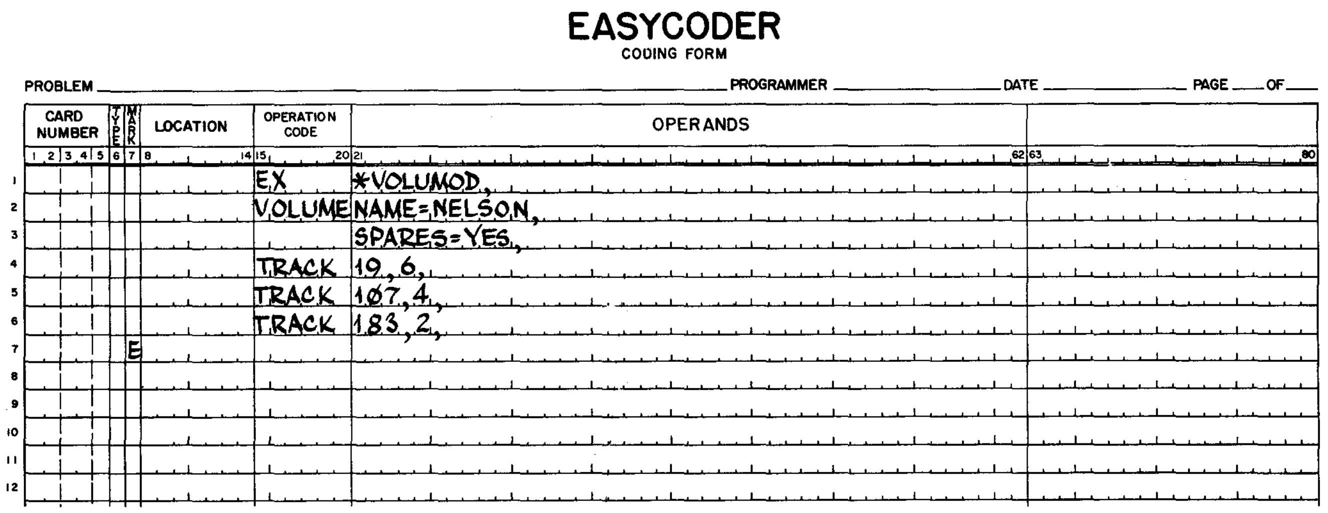
EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER 1 2 3 4 5	TYPE 6	MARK 7	LOCATION 8 14	OPERATION CODE 15 20	OPERANDS 21 62	63 80
1					EX	*VOLUMOD,	
2					VOLUME	NAME=SCOTTY,	
3						SPARES=YES,	
4						SERIAL=A12345,	
5			E				
6							
7							

In Example 1, Volume Modification D is run for the sole purpose of creating the *BADTRACKS and *VOLSPARES files in a standard area of a volume named SCOTTY, whose serial number is A12345. The volume name and serial number are to be checked before the two new files are created on the volume. (It is assumed that the *BADTRACKS and *VOLSPARES files have not been previously created on this volume by Volume Preparation C or an earlier run of Volume Modification D.)

EXAMPLE 2 - CREATING *BADTRACKS AND *VOLSPARES FILES, ASSIGNING SUBSTITUTES FOR BAD TRACKS



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER 1 2 3 4 5	TYPE 6	MARK 7	LOCATION 8 14	OPERATION CODE 15 20	OPERANDS 21 62	63 80
1					EX	*VOLUMOD,	
2					VOLUME	NAME=NELSON,	
3						SPARES=YES,	
4					TRACK	19,6,	
5					TRACK	1Ø7,4,	
6					TRACK	183,2,	
7			E				
8							
9							
10							
11							
12							

In Example 2, Volume Modification D is run to create the *BADTRACKS and *VOLSPARES files in a standard area of a volume named NELSON. The volume's name is to be checked but its serial number, if any, is not to be checked. After the two files have been created, a substitute track is assigned or established for the bad track specified in each of the three Track statements. (Any number of Track statements is permissible.)

EXAMPLE 3 - ASSIGNING SUBSTITUTES FOR BAD TRACKS (*BADTRACKS AND *VOLSPARES FILES PREVIOUSLY CREATED)

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					EX	*VOLUMOD,	
2					VOLUME	NAME=NELSON,	
3					TRACK	54, 2,	
4			E				
5							
6							
7							

In Example 3, Volume Modification D is run to assign or establish a substitute track for a bad track where address is cylinder 54, track 2. The bad track at that address has been discovered on the volume named NELSON at some time <u>after</u> Volume Modification D was run in Example 2. The *BADTRACKS and *VOLSPARES files were created in Example 2, so no parameter is needed to create them at this time. (Volume Modification D could also be run as shown in Example 3 if the *BADTRACKS and *VOLSPARES files had been previously created on the volume by Volume Preparation C.)

EXAMPLE 4 - LISTING BAD TRACKS RECORDED IN THE *BADTRACKS FILE

EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					EX	*VOLUMOD,	
2					VOLUME	NAME=MILNER,	
3			E				
4							
5							

In Example 4, Volume Modification D is run solely for the purpose of listing on the printer (1) the address of each bad track recorded in the *BADTRACKS file of a volume named MILNER and (2) the address of all substitute tracks on the volume.

The name of the volume is to be checked. The *BADTRACKS and *VOLSPARES files must have been previously created on MILNER either by Volume Preparation C or an earlier run of Volume Modification D.

Every run of Volume Modification D automatically produces a printer listing as described above. In this example it can be assumed that no newly discovered bad tracks exist since no Track statements are included.

JOB CONTROL LANGUAGE FOR VOLUME MODIFICATION D

To run Volume Modification D, a job control file, consisting of a series of job control statements, must be placed in the card reader. Each job control statement contains one or more parameters. Some job control statements are required for every run of Volume Modification D; others are optional.

In the following description of job control statements, the term "line" is the equivalent of one card in the job control file.

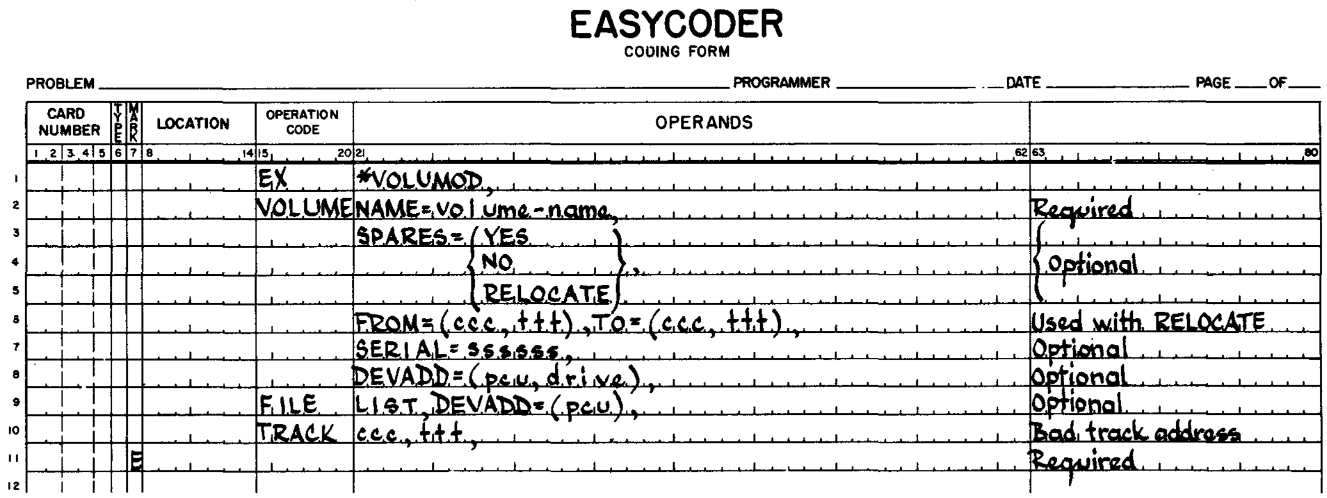
Sequence of Job Control Statements

Job control statements for Volume Modification D must appear in the following order:

1. Execute statement (if the routine is loaded from mass storage);
2. Volume statement;
3. File statement (if present); and
4. Track statement(s) (if present).

The last line of the last job control statement must contain an E in its mark field or be followed immediately by a line with blanks in its operation code and operands fields and an E in its mark field.

Figure 7-1 contains a complete set of job control statements (some of which are optional) for Volume Modification D.


EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					EX	*VOLUMOD,	
2					VOLUME	NAME=volume-name,	Required
3						SPARES={YES	{
4						NO},	Optional
5						RELOCATE}	{
8						FROM=(ccc, ttt), TO=(ccc, ttt),	Used with RELOCATE
7						SERIAL=ssssss,	Optional
8						DEVADD=(pcu, drive),	Optional
9					FILE	LIST, DEVADD=(pcu),	Optional
10					TRACK	ccc, ttt,	Bad track address
11			E				Required
12							

Figure 7-1. Job Control Statements for Volume Modification D

Execute Statement

The Execute statement is used only when Volume Modification D is loaded from mass storage. The Execute statement directs the Supervisor to load the Volume Modification D routine from the system disk. An Execute statement is required for every run of this routine when it is mass storage resident.

The Execute statement is not used when Volume Modification D is loaded from magnetic tape. (For a description of loading from magnetic tape, see "Operating Procedures for Volume Modification D," later in this section.)

Volume Statement

The Volume statement is always used to identify the name of the volume to be processed. The Volume statement also indicates the address (either user-specified or defaulted) of the disk device on which the named volume is mounted. In addition, the Volume statement can be used, at the user's option, to create the *BADTRACKS and *VOLSPARES files on the volume in either a standard area or a user-specified area. The parameters of the Volume statement are as follows:

VOLUME-NAME PARAMETER

The volume-name parameter (NAME) identifies the volume to be processed. This parameter is required for every run of Volume Modification D to insure that the correct volume is present. If an incorrect volume has been mounted, a halt or console message occurs. The form of this parameter is shown below.

```
NAME = volume-name,
```

volume-name = This name may be up to six characters in length. If fewer than six characters are specified, this field on the volume label is zero-filled to the left.

SPARES PARAMETER

The spares parameter can be used, at the user's option, to create the *BADTRACKS and *VOLSPARES files on the mass storage volume. If a *BADTRACKS file and a *VOLSPARES file have not been created on the volume by Volume Preparation C, they must be created by Volume Modification D before any processing can occur.

Unless the RELOCATE option is specified, the two files are created in a standard area on the volume. The RELOCATE option allows the user to locate the files in an area of his choice (with certain limitations, explained under "FROM and TO Parameters").

The form of this parameter is shown below:

$$\boxed{\text{SPARES} = \left\{\begin{array}{l}\text{YES}\\ \text{NO}\\ \text{RELOCATE}\end{array}\right\},}$$

YES = A *BADTRACKS file and a *VOLSPARES file are to be created. The two files will occupy all of the tracks in the following standard areas.

Device Type	Cylinders
155	200 - 202
258	101 - 103
259, 273, 259A, 259B	200 - 202
261, 262	125 - 127

NO = The *BADTRACKS and *VOLSPARES files are not to be created.

RELOCATE = A *BADTRACKS file and a *VOLSPARES file are to be created in an area specified by the user. This option may be used if these files cannot be created in the standard area. The RE-LOCATE option must be followed immediately by a pair of FROM and TO parameters.

Default assumption: NO

FROM AND TO PARAMETERS

A pair of FROM and TO parameters is used only when the RELOCATE option has been specified in the spares parameter. The pair of FROM and TO parameters enables the user to designate the area on the volume where the *BADTRACKS and *VOLSPARES files are to be created. The pair of FROM and TO parameters must immediately follow the spares parameter in the job control file.

The form of this parameter is shown below.

```
FROM = (ccc, ttt),   TO = ccc, ttt),
```

ccc = The cylinder address, in decimal, where leading zeros are not required. The first cylinder on the volume is numbered zero.

ttt = The track address, in decimal, where leading zeros are not required. The first track on each cylinder is numbered zero.

The pair of FROM and TO parameters specifies that the *BADTRACKS and *VOLSPARES files are to occupy the following area of the mass storage volume: all the tracks from the track stated in the FROM parameter to the track stated in the TO parameter (inclusive) on each cylinder from the cylinder stated in the FROM parameter to the cylinder stated in the TO parameter (inclusive).

The area designated by a pair of FROM and TO parameters can be as large as the user desires or as small as two tracks (one track for the *BADTRACKS file and one for the *VOLSPARES file). This user-designated area can be located anywhere on the volume except where it would conflict with the bootstrap track, substitute bootstrap track, volume label, volume directory, or an existing file.

VOLUME-SERIAL-NUMBER PARAMETER

This parameter is used to check the volume serial number (if any) written onto the volume label by Volume Preparation C. This parameter is used to insure that the correct volume is present. If an incorrect volume has been mounted, a halt or console message occurs. Use of this parameter is optional. The form of this parameter is shown below.

```
SERIAL = volume-serial-number,
```

volume-serial-number =	This value may be up to six alphanumeric characters in length. If fewer than six characters are specified, this field on the volume label is zero-filled to the left.

Default assumption: the volume serial number is not checked.

DEVICE-ADDRESS PARAMETER

The device-address parameter (DEVADD) indicates the address of the disk device holding the volume to be processed. The form of this parameter is shown below.

```
DEVADD = (pcu, drive),
```

pcu =	The peripheral control unit address of the disk device on which the volume is mounted, written as two octal characters. The I/O bit is not used, but all other bits, including the sector bits, must be specified.
drive =	The drive number, written as one octal character.

Default assumption: pcu 04, drive 0.

File Statement for the List File

During every run of Volume Modification D, the printer lists (1) the address of each bad track recorded in the *BADTRACKS file and (2) the addresses of all substitute tracks. This printer listing is called the list file.

The File statement for the list file is required only when the printer is assigned to an address other than peripheral control unit 02. The format of this statement is shown in line 9 of Figure 7-1.

DEVICE-ADDRESS PARAMETER

The device-address parameter (DEVADD) is used to indicate the peripheral control unit address of the printer. The form of this parameter is shown below.

```
DEVADD = (pcu),
```

pcu = The address of the printer's peripheral control unit. The address must be two octal characters. The I/O bit is not used, but all other bits, including sector bits, must be specified.

Default assumption: 02

Track Statement

A Track statement is used to indicate the address of a bad track for which a substitute is to be made during the current run of Volume Modification D. The format of the Track statement is shown on line 10 of Figure 7-1.[1]

A separate Track statement is required for each bad track. Any number of Track statements may be included, but they must follow all other job control statements of Volume Modification D.

Use of Track statements during the routine requires that the *BADTRACKS and *VOLSPARES files exist on the volume. (The two files may have been created previously or they may be created by the current run of the routine.)

OPERATING PROCEDURES FOR VOLUME MODIFICATION D

Loading Volume Modification D

Volume Modification D can be loaded in two ways:

1. From mass storage, by means of the Supervisor, under the Mod 1 (MSR) Operating System;
2. From magnetic tape, by means of Floating Tape Loader-Monitor C, under the Mod 1 (TR) Operating System.

MASS STORAGE RESIDENT

When the routine is loaded from mass storage, an Execute statement (as illustrated in Figure 7-1) must be submitted in the job control file, followed by the desired job control statements for the routine.

[1] The address of the bad track is written in the decimal format ccc, ttt, where ccc is the cylinder address and ttt is the track address. Leading zeros are not required.

TAPE RESIDENT

Loading the routine from magnetic tape is the same as loading from mass storage except that a console call card is used instead of an Execute statement. The format of the console call card is shown below.

Table 7-1. Console Call Card Format

Characters	Contents	Meaning
1 - 8	*VOLUMOD	Program and segment names of Volume Modification D.
9	d	Tape drive number from which Volume Modification D is to be loaded.
18	*	Asterisk (required by the Mod 1 (TR) Operating System).

Protection Switch Settings

Before running Volume Modification D, the user must set to PERMIT all protection switches on the peripheral control unit of the disk device holding the volume to be modified.

Operator Communication and Control

When an error condition occurs, 2-way communication between the operating system and the operator is achieved through a control panel <u>or</u> (if certain software and hardware requirements have been met) through a console. If the communication medium is a control panel, an error condition causes the central processor to halt. If the communication medium is a console, an error condition causes a typewriter message to appear, but the central processor does not halt.

Halts or console messages are caused by three types of error conditions: peripheral device conditions, job control file conditions, and conditions specific to Volume Modification D. When a halt or console message occurs, the operator must determine which error condition exists and then make the appropriate response.

CONTROL PANEL HALTS

At the time of a control panel halt, the B-address register contains a value that identifies the error condition. In most cases, the A-address register contains the address of a response location in main memory. The response location is a 1-character main memory location into which the operator must enter a response character appropriate to the error involved. Where indicated. additional information concerning the error can be found in the main memory locations immediately following the response location. This additional information is called the supplementary list (see Table 7-2) and is composed of the following elements (arranged in ascending order of contiguous main memory locations).

Table 7-2. Supplementary List

Field Name	Length
Specific code	One character (A+1)
Volume name	Six characters
Volume serial number	Six characters
Mass storage pcu address	One character
Device address	One character
Magazine number	One character
Error address[1]	Six characters in the decimal format cccttt
File name (if applicable)	Ten characters

[1]Significant only for certain mass storage peripheral device errors (otherwise blank).

Peripheral Device Halts

A B-address register value in the range from 0000 to 3777 indicates an error condition involving a peripheral device. The operator should consult Table 7-3. The B-address register value has the general form ppxd, where:

pp = Peripheral control unit (leftmost bit set to zero);

x = Code indicating type of error condition:

0 = Device not operable,

1 = Uncorrectable read error,

2 = Uncorrectable write error,

4 = Positioning error, and

7 = Miscellaneous error;

d = Device number.

These values are consistent with the standard peripheral error halt codes.

In all cases except the card read error, the A-address register contains the address of a response location. The supplementary list is also present (see Table 7-2).

Table 7-3. Peripheral Device Halts

B-Address Register Contents	Specific Code	Condition	Operator Action[1]

pp0d	01	Device inoperable.	G = retry.
			E = exit to the Supervisor.

	02	Protection violation.	

pp1d	07	Read error (mass storage).	G = retry.
			E = exit to the Supervisor.

		Read error (cards).	Correct erroneous card, refeed starting with that card, and press RUN.

pp2d	04	Format violation.	G = retry.
			E = exit to the Supervisor.

	12	Track overflow.	

	10	Write error.[2]	G = retry.
			S = skip defective track and resume processing.
			E = exit to the Supervisor.

pp4d	03	Positioning error.	G = retry.
			E = exit to the Supervisor.

	13	Positioning error.	E = exit to the Supervisor.

pp7d	11	Miscellaneous device error condition - possible device failure.	G = retry.
			E = exit to the Supervisor.

[1]The operator enters the appropriate response character and presses RUN.

G = 27_8, S = 62_8, E = 25_8.

[2]An unusable track has been encountered on the volume. The printer produces the following message: CYL ccc TR ttt UNUSABLE (ccc and ttt are decimal numbers). "Unusable" means that the track surface is bad and not even one bad-track track-linking record can be read. It is very unlikely that any track will ever be unusable.

Job Control File Halts

A B-address register value in the range from 5000 to 5077 signifies an error condition involving the job control file. The operator should consult Table 7-4. The A-address register contains the address of the response location. The supplementary list is not produced.

Table 7-4. Job Control File Halts

B-Address Register Contents	Condition	Operator Action[1]
5001	Invalid command field.	G = attempt to perform the operation again. The operator corrects the erroneous job control statement; refeeds the cards,
5002	Invalid positional parameter.	
5003	Invalid keyword.	

Table 7-4 (cont). Job Control File Halts

B-Address Register Contents	Condition	Operator Action[1]
5004	Required parameter missing.	starting with the Volume statement; enters a G; and presses RUN. The program then reprocesses the job control file.
5005	Invalid keyword parameter value.	
5010	Invalid combination or sequence of parameters.	E = exit to the Supervisor.
[1]G = 27_8, E = 25_8.		

Halts Specific to Volume Modification D

A B-address register value in the range from 6550 through 6567 indicates a halt specific to Volume Modification D. The operator should consult Table 7-5.

The A-address register contains the address of the response location. The supplementary list is also present (see Table 7-2).

Table 7-5. Halts Specific to Volume Modification D

B-Address Register Contents	Specific Code	Condition	Operator Action[1]
6550		Wrong volume mounted:	G = reread the volume label.
	01	Volume-name check failed.	E = exit to the Supervisor.
	02	Volume-serial-number check failed.	
6551		An attempt is being made to process a Track statement, but the *BADTRACKS and *VOLSPARES files have not been created on the volume.	E = exit to the Supervisor.
6552		Either of two conditions exists: 1. There are no more available substitutes in the *VOLSPARES file. Further track substitution is possible only for a Track statement containing	S = skip the current Track statement and continue to the next Track statement, if any. E = exit to the Supervisor.

Table 7-5 (cont). Halts Specific to Volume Modification D

B-Address Register Contents	Specific Code	Condition	Operator Action[1]
6552 (cont)		the address of the bootstrap track or the substitute bootstrap track. 2. There is no substitute bootstrap track available. (The volume cannot be used as a system disk.)	
6553	01	A bad track exists in a file that does not meet the requirements for the use of substitute tracks.	S = skip the current Track statement and continue to the next Track statement, if any. E = exit to the Supervisor.
	02	A substitute track, currently established for a bad track in a partitioned sequential file, has been declared bad in a Track statement.	A = establish a new substitute track and continue processing. S and E are the same as for 01.
6554		A bad track exists in the *BADTRACKS file. (Volume Modification D must be run again with the RELOCATE option.)	E = exit to the Supervisor.
6555		There is an unusable[2] track in the volume label or the volume directory. (The volume cannot be used.)	E = exit to the Supervisor.
6556		The bootstrap track (cylinder 000, track 000) is unusable.[2] (The volume cannot be used as a system disk; i.e., it cannot be bootstrapped.)	A = accept the condition and continue processing the volume. E = exit to the Supervisor.
6557		Duplication error:	E = exit to the Supervisor.
	01	An attempt is being made to create the *BADTRACKS and *VOLSPARES files and these files already exist on the volume.	
	02	The unit of allocation (either specified or defaulted) for the *BADTRACKS and *VOLSPARES	

Table 7-5 (cont). Halts Specific to Volume Modification D

B-Address Register Contents	Specific Code	Condition	Operator Action[1]
6557 (cont)		files conflicts with that of a file already existing on the volume.	
	03	A substitute has already been assigned for the bad track named in the current Track statement.	S = skip the current Track statement and continue to the next Track statement, if any. E = exit to the Supervisor.
6567		There is insufficient memory for the buffer required by the current Track statement.	S = skip the current Track statement and continue to the next Track, statement, if any. E = exit to the Supervisor.

[1]The operator enters the appropriate response character and presses RUN ($G = 27_8$, $S = 62_8$, $A = 21_8$, $E = 25_8$).

[2]"Unusable" means that the track surface is bad and not even one bad-track TLR can be read from the track. It is very unlikely that any track will ever be unusable.

CONSOLE MESSAGES

The console can be used for operator control when the Supervisor (in the Mod 1 (MSR) Operating System) or Floating Tape Loader-Monitor C (in the Mod 1 (TR) Operating System) has been appropriately specialized.

When the console is used for operator control, an error condition does not cause the system to halt. Instead, a diagnostic message is typed out on the console. The operator takes the following steps.

1. He reads the typeout. (To repeat the message, he presses the space bar twice.) If necessary, he consults the manual for possible action.
2. He performs the desired corrective action.
3. He types the appropriate 1-character response (G, E, etc.).
4. If the typein is correct, he presses the space bar to continue. If it is incorrect, he types any nonspace character and returns to step 3.

The routine continues as directed by the response code. If an invalid response is made, the routine causes the diagnostic message to reappear.

The following paragraphs provide detailed information about the three kinds of console messages.

Peripheral Device Messages

Console messages specific to peripheral device error conditions are listed in Table 7-6. The format of these messages is:

pp d MESSAGE TEXT

where pp = Peripheral control unit (leftmost bit set to zero),

d = Device number, and

MESSAGE TEXT specifies the error condition.

A second line of typed information contains the supplementary list described in Table 7-2.

Table 7-6. Peripheral Device Messages

Typewriter Message	Specific Code (alpha-numeric)	Condition	Operator Action[1]

pp d INOPERABLE	1
	2
		Device inoperable.
		Protection violation.
			G = retry.
			E = exit to the Supervisor.

pp d READ ERROR	7	Read error (mass storage).	G = retry.
			E = exit to the Supervisor.

		Read error (cards).	Correct erroneous card, refeed starting with that card, and press space bar.

pp d WRITE ERROR	4
	'
		Format violation.
		Track overflow.
			G = retry.
			E = exit to the Supervisor.

	8	Write error.[2]	G = retry.
			S = skip defective track and resume processing.
			E = exit to the Supervisor.

pp d POSITIONING ERROR	3	Positioning error.	G = retry.
			E = exit to the Supervisor.

	=	Positioning error.	E = exit to the Supervisor.

pp d MISCELLANEOUS	9	Miscellaneous device error condition - possible device failure.	G = retry.
			E = exit to the Supervisor.

[1]The operator types the appropriate alphabetic response character and presses the space bar.

[2]An unusable track has been encountered on the volume. The printer produces the following message: CYL ccc TR ttt UNUSABLE (ccc and ttt are decimal numbers). "Unusable" means that the track surface is bad and not even one bad-track track-linking record can be read. It is very unlikely that any track will ever be unusable.

Job Control File Messages

All messages involving errors in the job control file begin with the words JOB CONTROL FILE ERROR. Table 7-7 contains the complete texts of all job control file messages for Volume Modification D. The supplementary list is not produced.

Table 7-7. Job Control File Messages

Typewriter Message	Condition	Operator Action[1]
JOB CONTROL FILE ERROR, COMMAND FIELD	Invalid command field.	G = attempt to perform the operation again. The operator corrects the erroneous job control statement; refeeds the cards, starting with the Volume statement; enters a G; and presses the space bar. E = exit to the Supervisor.
JOB CONTROL FILE ERROR, POSITIONAL PARAMETER	Invalid positional parameter.	
JOB CONTROL FILE ERROR, KEYWORD PARAMETER	Invalid keyword.	
JOB CONTROL FILE ERROR, MISSING PARAMETER	Required parameter missing.	
JOB CONTROL FILE ERROR, PARAMETER VALUE	Invalid keyword parameter value.	
JOB CONTROL FILE ERROR, PARAMETER COMBINATION	Invalid combination or sequence of parameters.	

[1]The operator types the appropriate alphabetic response character and presses the space bar.

Messages Specific to Volume Modification D

Table 7-8 contains messages for error conditions specific to Volume Modification D. The typewriter messages in Table 7-8 are followed by a second line of typed information, which contains the supplementary list described in Table 7-2.

Table 7-8. Messages Specific to Volume Modification D

Typewriter Message	Specific Code (alphanumeric)	Condition	Operator Action[1]
INCORRECT VOLUME		Wrong volume mounted:	G = reread the volume label. E = exit to the Supervisor.
	1	Volume-name check failed.	
	2	Volume-serial-number check failed.	
*BADTRACKS NOT ON VOLUME		An attempt is being made to process a Track statement, but the *BADTRACKS and *VOLSPARES files have not been created on the volume.	E = exit to the Supervisor.

Table 7-8 (cont). Messages Specific to Volume Modification D

Typewriter Message	Specific Code (alphanumeric)	Condition	Operator Action[1]
NO SUBSTITUTE TRACK AVAILABLE		Either of two conditions exists: 1. There are no more available substitutes in the *VOLSPARES file. Further track substitution is possible only for a Track statement containing the address of the bootstrap track or the substitute bootstrap track. 2. There is no substitute bootstrap track available. (The volume cannot be used as a system disk.)	S = skip the current Track statement and continue to the next Track statement, if any. E = exit to the Supervisor.
FILE CANNOT CONTAIN BAD TRACKS	1	A bad track exists in a file that does not meet the requirements for the use of substitute tracks.	S = skip the current Track statement and continue to the next Track statement, if any. E = exit to the Supervisor.
	2	A substitute track, currently established for a bad track in a partitioned sequential file, has been declared bad in a Track statement.	A = establish a new substitute track and continue processing. S and E are the same as for 1.
BAD TRACK IN *BADTRACKS FILE		A bad track exists in the *BADTRACKS file. (Volume Modification D must be run again with the RELOCATE option.)	E = exit to the Supervisor.
UNUSABLE TRACK IN DIRECTORY OR LABEL		There is an unusable[2] track in the volume directory or the volume label. (The volume cannot be used.)	E = exit to the Supervisor.

Table 7-8 (cont). Messages Specific to Volume Modification D

Typewriter Message	Specific Code (alphanumeric)	Condition	Operator Action[1]
UNUSABLE BOOT-STRAP TRACK		The bootstrap track (cylinder 000, track 000) is unusable.[2] (The volume cannot be used as a system disk; i.e., it cannot be bootstrapped.)	A = accept the condition and continue processing the volume. E = exit to the Supervisor.
DUPLICATION ERROR		Duplication error:	E = exit to the Supervisor.
	1	An attempt is being made to create the *BADTRACKS and *VOLSPARES files and these files already exist on the volume.	
	2	The unit of allocation (either specified or defaulted) for the *BADTRACKS and *VOLSPARES files conflicts with that of a file already existing on the volume.	
	3	A substitute has already been assigned for the bad track named in the current Track statement.	S = skip the current Track statement and continue to the next Track statement, if any. E = exit to the Supervisor.
INSUFFICIENT MEMORY FOR BUFFERS		There is insufficient memory for the buffer required by the current Track statement.	S = skip the current Track statement and continue to the next Track statement, if any. E = exit to the Supervisor.

[1]The operator enters the appropriate alphabetic response character and presses the space bar.

[2]"Unusable" means that the track surface is bad and not even one bad-track TLR can be read from the track. It is very unlikely that any track will ever be unusable.

APPENDIX A
CALCULATION OF SORT-ITEM BLOCK SIZE

Parameter 08 of the Fetch macro routine specifies the buffer size required for reading blocks from the sort-item file. If the value given to this parameter is less than that required, the Fetch cannot be executed. If, however, the assigned value of the parameter exceeds the requirement, some memory locations will be wasted, but the Fetch will be executed. Also, calculation of the sort-item block size is necessary for an accurate computation of the work area required to execute the sort.

The calculation of the sort-item block size is based upon certain previous calculations, which are described below. Certain other information is derived as a result of these calculations such as the maximum sort-item size acceptable to the sort.

CALCULATING HIGHEST MEMORY LOCATION AVAILABLE TO PRESORT

No Presort Own-Coding Present

Let HMPS = HMA, where HMA represents the highest memory address available to the sort. (Note that HMPS is not always equal to the value specified by the user for the highest memory location.)

Presort Own-Coding Present

When own-coding is present, the highest memory address available to any logical segment of the sort (i.e., presort or merge) may be determined from the base address of the own-code program that is resident during the execution of that logical segment.

OWN-CODING OUTSIDE SORT AREA

If all own-code values are greater than the highest memory address parameter or lie below the base of the sort, then let HMPS = HMA. (Note that own-coding can only lie below the base of the sort if the sort has been relocated from its present base value of 200 decimal.)

OWN-CODING WITHIN SORT AREA

If there is no merge own-coding or the merge own-code program is to be called by the sort, the following applies.

1. PSOPEN is defined as the value of the presort open exit.
2. PSITEM is defined as the value of the presort item exit.
3. Let HMPS = PSOPEN-1, if PSOPEN is less than PSITEM; otherwise, let HMPS = PSITEM-1.

When merge own-coding is loaded prior to the execution of the sort, then the following applies.

1. MERGE is defined as the value of the merge own-code exit.
2. HMPS is defined as the least of the three values PSOPEN-1, PSITEM-1, or MERGE-1.

CALCULATING HIGHEST MEMORY LOCATION AVAILABLE TO MERGE

Merge own-code is active only during the merge multicylinder phase; and therefore, the merge 1-cylinder phase is affected only if the merge own-code is resident at the beginning of sort execution. Calculation of HMMC is significant only in terms of the merge multicylinder phase.

No Merge Own-Coding Present

The formula HMMC = HMA applies.

Merge Own-Coding Present

OWN-CODING OUTSIDE SORT AREA

If the merge own-coding value is greater than the highest memory address parameter or lies below the base of the sort, then the formula HMMC = HMA applies.

OWN-CODING WITHIN SORT AREA

The formula HMMC = (MERGE-1) applies.

CALCULATING SORT-ITEM SIZE

The following variables are defined for calculating sort-item size:

KL = Total number of key characters;

EL = Total number of extract characters;

AS = Zero, if no item address is appended; 11, if item address is appended; and

SIS = KL + EL + AS.

If an item sort is requested and IL represents the input item length, then the following formula applies.

SIS = IL + AS

In general, for an item sort, it is expected that AS = 0.

CALCULATING SPACE AVAILABLE TO PRESORT

For calculating the space available to presort, the following formula applies.

PS = HMPS - PROG - SUP - TYPE,

where

SUP	=	Supervisor memory requirement (see Supervisor manual);
TYPE	=	2150 if a typewriter is present, otherwise 0;
PROG	=	7650 when sequential files are processed;
PROG	=	8750 when direct access files are processed;
PROG	=	8750 when indexed sequential files are processed;
PROG	=	7650 when input to the sort is from 1/2-inch magnetic tape; and
PROG	=	6000 when input to the sort is through own-coding only or when input is from cards.

CALCULATING SPACE AVAILABLE TO MERGE

The following formula applies for calculating the space available to the merge:

PM = HMMC - 5700 - SUP - 2(SIS) - TYPEM,

where: TYPEM = 1500 if a typewriter is present, otherwise 0.

Single and Double Buffering Mode

Whether to use single or double buffering is decided at presort execution time. This decision is significant in determining the sort-item block size (see Tables A-1 and A-2). If bad tracks are present in any of the volume areas used for the work files, use Tables B-4 and B-5 for all calculations involving the disk tables (see "Bad Track Handling Procedures for Mass Storage Sort C" in Appendix B).

Table A-1. Disk Table (Type 155, 258, 259, 273, 259A and 259B Disk Pack Drives)

Column A	Column B
Data Characters Per Block	Records Per Block
241	1
491	2
741	3
991	4
1,241	5
1,491	6
2,491	10
3,741	15
NOTE: Use Table B-4 if bad tracks are present in work file(s).	

Table A-2. Disk Table (Type 261 and 262 Disk Files)

Column A	Column B
Data Characters Per Block	Records Per Block
241	1
491	2
741	3
991	4
1,241	5
1,491	6
2,491	10
3,741	15
4,991	20
7,491	30
NOTE: Use Table B-5 if bad tracks are present in work file(s).	

When the work files are on Types 155, 258, 259, 273, 259A, or 259B Disk Pack Drives, use Table A-1, but if the work files are on Type 261 or 262 Disk Files, use Table A-2 to find the least value in column A that is greater than or equal to the sort-item size (SIS). Let that value be represented by PBS (physical block size).

1. Calculate $NI_1 = \left[\frac{PBS}{SIS} \right]$

   where: NI_1 = temporary value for the sort-item block factor (rounded down to the next lowest integer).

2. Calculate $IS_1 = 3 \left[NI_1 \ (SIS + 12) \right]$

   where: IS_1 = a temporary item storage factor.

3. Calculate $OS_1 = NI_1 \left[SIS \right]$

   where: OS_1 = a temporary value for the block size.

The presort will operate with double buffering if the two following conditions are met viz:

1. $PS \geq 2\,(INB) + IS_1 + 2(OS_1) + 26$, and
2. $PM \geq 3(OS_1) + 26$.

CALCULATING SORT-ITEM BLOCK SIZE FOR SINGLE BUFFER MODE

To determine the sort-item block size for the single buffering mode, perform the following calculations:

1. Calculate $NI_2 = \left[\frac{PS - INB - 13}{2\,(SIS + 6)} \right]$ (rounded down to the next lower integer), and
2. Calculate $NI_3 = \left[\frac{PM - 34}{SIS} \right]$ (rounded down to the next lower integer).

NI is the least of the three values NI_1, NI_2, or NI_3, where NI represents the sort-item blocking factor (NI_1 having been calculated in the previous paragraph). The sort-item block size for single buffering, then, is:

$$OS = NI\ (SIS) + 9$$

CALCULATING SORT-ITEM BLOCK SIZE FOR DOUBLE BUFFER MODE

When the presort finds that double buffering is possible, it attempts to increase the sort-item block size, provided that a string of items can be maintained that holds at least three times the number of items that can be contained in a sort-item block. A further factor in the computation is that there should be an efficient covering of the work-file area.

To determine the sort-item block size for the double buffering mode, perform the following.

1. Calculate $PMD = PS - 2\ (INB) - 26$

   where: PMD = the space available to the presort after an allowance has been made for the two input buffers and output buffer control.

2. Calculate $ISB = 5\ (SIS) + 36$

   where: ISB = an item capacity unit.

3. Calculate $NI_1 = \frac{PMD}{ISB}$ (rounded down to the next lowest integer)

4. Calculate $OS_1 = NI_1\ (SIS)$.

5. Using Table A-1 or Table A-2, find in column A the least value greater than or equal to OS_1. Let that value be PBS_h. If there is a member of Table A-1 or Table A-2 less in value than PBS_h, represent it as PBS_L. This rule has two exceptions, as shown below.

   a. If $PBS_h = 241$, go to step 8c,

   b. If OS_1 is greater than 3,741, calculate the following:

   $NI = \left[\frac{3741}{SIS}\right]$ (rounded down to next lowest integer); go to step 9.

6. Calculate the following:

   $W_h = PBS_h - NI_1\ (SIS)$

   where: W_h = the number of characters of a group of physical records represented by PBS_h that would not be occupied by a sort-item block.

   NI_1 is calculated in step 3.

7. Calculate the following:

   a. $NI_2 = \left[\frac{PBS_L}{SIS}\right]$ (rounded down to the next lowest integer)

   b. $W_L = PBS_L - NI_2\ (SIS)$

   where: W_L = the number of characters of the group of physical records that would not be occupied by the sort-item block.

8. Calculate the following:

   a. If $W_h \leq W_L$ then $NI = NI_1$;

   b. If $W_h \geq W_L$ then $NI = NI_2$; and

   c. When $PBS_h = 241$, $NI = \left[\frac{241}{SIS}\right]$ (rounded down to the next lowest integer)

9. The sort-item block size = OS = NI (SIS) + 9.

APPENDIX B
BAD TRACK HANDLING PROCEDURES

This appendix contains a general description of bad track handling procedures incorporated in the Mod 1 (MSR) Operating System. Following the general description, there is information regarding bad track handling procedures applicable to three utility routines: Volume Preparation C, Mass Storage Sort C, and Disk/Tape Copy C.

TRACK SUBSTITUTION CAPABILITY

The Mod 1 (MSR) Operating System provides the user with the option of track substitution capability. Track substitution is designed to minimize inconvenience and possible data loss that might result from an uncorrectable read error caused by a defect on the magnetic surface of a mass storage volume. A track on which an uncorrectable read error occurs is defined as a "bad track."

In general, whenever a bad track is encountered by the system software, the operator is notified by a control panel halt or a console message.[1] A spare track can then be substituted for the bad track and file data (if any) transferred from the bad track to the substitute track; in addition, when data is transferred, the contents and header area of any record for which a read error was detected are printed out, as accurately as possible, for the user's information. Once a spare track has been established as the substitute for a bad track, the system's logical and physical I/O control can subsequently process the file containing that bad track, automatically using the substitute track instead.

Advantages of Track Substitution

The methods of dealing with bad tracks prior to track substitution capability were often time consuming. The file in which a bad track occurred had to be unloaded (if possible), deallocated, reallocated so as not to include the bad track, and finally reloaded. A bad track in a critical area caused special difficulty. If the bootstrap track was bad, the volume could not be used as a system residence volume (because it could not be bootstrapped). If either the volume label track or any track in the volume directory was bad, the volume could not be used at all. With track substitution capability, however, the occurrence of a bad track is of little significance to the usability of a volume.

[1] Exceptions to this rule are noted in documentation of individual system programs.

Preparing the Volume for Track Substitution

Bad track handling capability is an integral part of the operating system software, but its use is dependent on the method of preparation of the individual mass storage volume. In order to substitute spare tracks for bad tracks on a volume, the volume must have been processed to create that capability, either initially by Volume Preparation C or by Volume Modification D at some time after initial volume preparation.

THE TRACK SUBSTITUTION FILES: *BADTRACKS AND *VOLSPARES

When the track substitution capability has been created on a mass storage volume, that volume differs from other volumes in that it contains two additional files: the bad tracks file (named *BADTRACKS) and the spare tracks file (named *VOLSPARES). The *BADTRACKS file contains a permanent list of all bad tracks ever encountered and declared on the volume. The *VOLSPARES file consists of all the tracks available for use as substitutes for bad tracks.

Each data item in the *BADTRACKS file contains the address of a bad track and the address of its assigned substitute track in the*VOLSPARES file, as well as certain status information. The two files are normally allocated together in a single unit of allocation of standard size and location by Volume Preparation C or Volume Modification D, although, if necessary, the user can select a different size or location for that unit of allocation.

*BADTRACKS File

The *BADTRACKS file is a standard sequential file. Once it is created by Volume Preparation C or Volume Modification D, it can be deallocated, allocated, loaded, or unloaded like other sequential files. The user, however, need not ordinarily process the *BADTRACKS file directly, since it is used primarily for reference by the system software. For example, after each run of Volume Modification D, for whatever purpose, the user receives a printer listing of all known bad tracks on the volume and their assigned substitutes, as derived from the latest updated contents of the *BADTRACKS file.

*VOLSPARES File

The *VOLSPARES file is a file of nonstandard organization. It cannot be processed by file-oriented programs, including File Support C and Logical I/O C. Since each utilized track in the *VOLSPARES file is a substitute for a bad track in some other file, the contents of the *VOLSPARES file are fully accessible by logical references within the files containing the bad tracks for which the utilized tracks were originally substituted.

Encountering a Bad Track

When an uncorrectable read error occurs on a mass storage volume, the system has by definition encountered a bad track. The response of the software varies from program to program and also depends on whether the user has elected to include track substitution capability on the volume involved.

In an environment without track substitution capability (that is, on a volume for which no *BADTRACKS and *VOLSPARES files have been created) the system executes a standard peripheral device condition halt or console message to indicate a read error, the peripheral control unit number, and the device number. In addition, the cylinder and track address of the bad track is available to the user in the supplementary list. (In Volume Preparation C, a write error is indicated, and the printer lists the message: CYL nnn TR nnn ERROR.) See Table B-1 for the location of the bad track address in the supplementary list.

When a bad track is encountered in an environment where track substitution capability has been elected, a substitute track can be assigned or established for the bad track. A substitute track is <u>assigned</u> to a bad track that is <u>not</u> part of a file on the mass storage volume. A substitute track is <u>established</u> for a bad track that is part of a file on the mass storage volume. Volume Preparation C automatically <u>assigns</u> a substitute track from the *VOLSPARES file, updates the *BADTRACKS file, and continues surface checking the newly formatted volume; at the end-of-run, the printer lists the addresses of any bad tracks encountered and all substitute tracks.

When a bad track is encountered during allocation of a file, and track substitution capability exists, the allocate function automatically <u>establishes</u> a substitute track for the bad track; the process of establishing a substitute track is defined and described below under the next heading.

When a bad track is encountered while running other system programs or user programs together with Logical I/O C, a peripheral device condition halt or console message is produced to indicate a read error, the peripheral control unit number, and the device number. In addition, the address of the bad track is contained in the supplementary list. The cylinder, track, and record address is given as three 2-character binary numbers. (For Volume Preparation C, Volume Modification D, and Disk/Tape Copy C, the bad track address is given in two 3-character decimal numbers indicating cylinder and track.)

Table B-1. Mass Storage Peripheral Device Error Information (Supplementary list begins at location A+1. Cylinder and track address of bad track is shaded.)

A-Address Register Location	Volume Preparation C	Volume Modification D	Mass Storage Sort C	Disk/Tape Copy C	Logical I/O File Support & Pro-Gram Develop-ment Subsystem	A-Address Register Location
A	response	response	response	response	response	A
A + 1	specific code	specific code	specific code	specific code	specific code	A + 1
A + 2			pcu			A + 2
A + 3			device			A + 3
A + 4	volume name	volume name	pack	input volume name		A + 4
A + 5	6 characters	6 characters	**cylinder**	6 characters	file name	A + 5
A + 6	(alphanumeric)	(alphanumeric)	**2 characters (binary)**	(alphanumeric)	10 characters (alphanumeric)	A + 6
A + 7			**track**			A + 7
A + 8			**2 characters (binary)**	device type code		A + 8
A + 9	volume serial number	volume serial number	record	pcu		A + 9
A + 10	6 characters (alphanumeric)	6 characters (alphanumeric)	2 characters (binary)	device		A + 10
A + 11				pack		A + 11
A + 12					relative volume number	A + 12
A + 13						A + 13
A + 14	pcu	pcu		output volume name	volume name	A + 14
A + 15	device	device		6 characters (alphanumeric)	6 characters (alphanumeric)	A + 15
A + 16	pack	pack				A + 16
A + 17	**cylinder**	**cylinder**				A + 17
A + 18	**3 characters (decimal)**	**3 characters (decimal)**		device type code		A + 18
A + 19				pcu	pcu	A + 19
A + 20	**track**	**track**		device	device	A + 20
A + 21	**3 characters (decimal)**	**3 characters (decimal)**		pack	pack	A + 21
A + 22				**cylinder**	**cylinder**	A + 22
A + 23				**3 characters (decimal)**	**2 characters (binary)**	A + 23
A + 24		file name (if any)			**track**	A + 24
A + 25		10 characters (alphanumeric)		**track**	**2 characters (binary)**	A + 25
A + 26				**3 characters (decimal)**	record	A + 26
A + 27					2 characters (binary)	A + 27
Etc.				file name (if any) 10 characters (alphanumeric)		Etc.

Establishing a Substitute Track in a File

When a bad track occurs within the area of allocation for a file, the spare track substituted for it is said to be established. Establishing a substitute is accomplished automatically at allocation time by File Support C, or by Volume Modification D after a file has been allocated.

Establishment of a substitute track in a file involves (1) linking the substitute track to the track that follows the bad track in the file; (2) formatting the substitute track in accordance with the file's allocation parameters; (3) entering the name of the file containing the bad track, and the addresses of the bad track and the substitute track, into an item in the *BADTRACKS file; (4) writing onto the bad track a bad-track track-linking record (TLR), which points to the substitute track; and (5) setting a field in the volume directory (character position 15 of the file's *VOLDESCR* entry) to indicate the presence of bad tracks in the file.

A substitute track can be established by the allocate function of File Support C when a bad track is encountered during allocation or by Volume Modification D when the user submits a Track statement declaring the cylinder and track address of each bad track encountered in a file after allocation.

When Volume Modification D establishes a substitute track, it also transfers all records containing file data from the bad track to the sustitute track; in addition, the header area and contents of any record for which a read error is detected are printed out as accurately as possible for the user's inspection. At the end-of-run, Volume Modification D also produces a printer listing of all known bad tracks and all substitute tracks on the volume.

During allocation, of course, there is no data to be transferred from a bad track to its substitute, since the file has not yet been loaded.

All software of the Mod 1 (MSR) Operating System can process any file containing bad tracks if substitutes have been established for them by the allocate function or by Volume Modification D.

FILE ORGANIZATION RESTRICTIONS ON ESTABLISHING SPARE TRACKS

In order to establish a substitute for a bad track, the file containing the bad track must have been allocated with an integral number of blocks per track. In addition, if a bad track is encountered in an indexed sequential file, no substitute can be established for it if the bad track is (1) the last track of the general overflow area, (2) the last track of the cylinder overflow area of any cylinder, or (3) the last track of the prime data area of any cylinder. If any of these restrictions applies, the bad track is not assigned a substitute track, the substitute is not established in the file, and a halt or console message occurs. (If control panel I/O is used, the

B-address register value 5455 appears during allocate and the value 6553 appears during Volume Modification D. If console I/O is used, the following message appears for both programs: FILE CANNOT CONTAIN BAD TRACKS.) The halt or console message indicates that the file cannot have a substitute track established for the bad track without reallocation. If this condition occurs during allocation, the file must be deallocated before any further processing of the volume(s) involved.

DEALLOCATING A FILE AFTER SUBSTITUTE TRACKS HAVE BEEN ESTABLISHED

When a file containing bad tracks with established substitute tracks is deallocated, the bad tracks remain assigned to their respective substitute tracks. However, the file's name is removed from each entry in the *BADTRACKS file and the status field in those entries is set to indicate that the bad track is not contained within a file.

BAD-TRACK TRACK-LINKING RECORD (TLR)

When a substitute track is established for a bad track, the Volume Modification D or allocate program writes at least one readable bad-track TLR onto the bad track, linking it to the substitute track. Thus, subsequent references to the bad track will be automatically routed to the established substitute track by the system programs. In the event of gross physical abuse to the magnetic surface of a disk during handling or storage, it is conceivable that no readable bad-track TLR could be written. It is highly unlikely that this situation will ever occur during the normal life of any mass storage volume; if it should, however, a standard peripheral device write error halt or console message results and the printer produces the following message: CYL nnn TR nnn UNUSABLE.

CONSIDERATIONS IN PROCESSING A VOLUME USING TRACK SUBSTITUTION

The user gains the advantages of track substitution capability without having to make fundamental alterations in his programs or previous operating procedures. In some cases, the following suggestions, if followed, will prove helpful.

Direct Access Files

Some difficulty may arise from the use of absolute bucket addressing in a direct access file that contains bad tracks. The mass storage address that can be obtained from the Logical I/O C communications area may be the address of a substitute track and, if so, cannot be returned to Logical I/O C as the absolute address of a bucket in the file.

Files Containing Mass Storage Address Information

Occasionally, file items include fields that contain the mass storage address of another item in the file or of another file. If track substitution capability is to be used on files of this type, the addresses of these items should be calculated rather than retrieved from the mass storage address register.

Substitute Bootstrap File

If the bootstrap track (cylinder 000, track 000) of a volume is bad, a substitute bootstrap file can be created by a Track statement to Volume Preparation C or Volume Modification D. Volume Preparation C automatically creates the substitute bootstrap file if the bootstrap track is found to be bad during initial volume preparation. The substitute bootstrap file occupies a single track on cylinder 000. If a substitute bootstrap file is established by a Track statement on an existing system volume, Bootstrap Generator C should be rerun to insure the validity of the bootstrap information.

Backup Procedures for the *BADTRACKS File

The information contained in the *BADTRACKS file is critical to the continued successful processing of the mass storage volume and must therefore be protected against possible data loss, as could occur for example if a track in the *BADTRACKS file itself became bad. The *BADTRACKS file cannot, by system definition, contain any bad tracks. The following paragraphs describe measures that can be taken to protect the highly critical contents of the *BADTRACKS file.

UNLOADING THE *BADTRACKS FILE

The *BADTRACKS file can be unloaded by File Support C to any storage medium available to that program. Backup of the contents of the file can be obtained by periodically unloading the file to tape or cards. (If this backup procedure were performed after each run in which a change in the contents of the *BADTRACKS file has occurred, the data on the backup medium would possess 100 percent validity at all times.)

RE-CREATING THE *BADTRACKS FILE

If a track in the *BADTRACKS file becomes bad, the file must be moved to another area of the mass storage volume where good tracks exist. The *BADTRACKS file itself normally occupies not more than two tracks on the volume, using the standard size unit of allocation. (The *VOLSPARES file occupies the bulk of the three cylinders allotted to the track substitution files in the standard unit of allocation.)

Deallocating the *BADTRACKS File

The *BADTRACKS file can be deallocated in exactly the same manner as any data file on the volume. Before deallocation is performed on this file, however, a halt or console message occurs to prevent inadvertent removal of the file. An appropriate response can be made to direct the system to perform the deallocation.

Allocating the *BADTRACKS File

The *BADTRACKS file is always initially created (at the same time as *VOLSPARES) by either Volume Preparation C or Volume Modification D. It can be re-created, following deallocation, by the allocate function. There are several special considerations when allocating *BADTRACKS, as follows:

1. The *VOLSPARES file must already exist on the volume.
2. The file name *BADTRACKS must be specified in the File statement.
3. The file organization specified in the File statement must be "sequential."
4. No password may be specified.
5. None of the parameters of the Size statement may be specified.
6. Only one pair of FROM and TO parameters (one unit of allocation) may be specified in the Units statement.
7. Only one Units statement may be specified.

If any of the above requirements is not met, a halt or console message occurs and allocation will not be performed. The appropriate values for item size and record size for the *BADTRACKS file will be set up by File Support C.

Loading the *BADTRACKS File

After the *BADTRACKS file has been moved to another area of the mass storage volume by use of the deallocate and allocate functions, the data on the backup storage medium can be loaded into the file by the load function of File Support C.

Prior to actually loading data into the *BADTRACKS file, a halt or console message occurs to warn the operator and prevent inadvertent destruction of the file's data. An appropriate response can be made to direct the system to continue and to load the file.

The Map Function

When using the DESCR or EXPIRED options of the File Support C map function, the printed description produced for each file on the volume indicates whether that file contains bad tracks.

BAD TRACK HANDLING PROCEDURES FOR VOLUME PREPARATION C

The user must give special attention to the subject of bad tracks on a mass storage volume that is being prepared by Volume Preparation C. Before Volume Preparation C is run, the user may be aware that one or more bad tracks exist on the volume to be prepared. In addition, while Volume Preparation C is formatting the volume in question, one or more bad tracks (previously unknown) may be encountered on the volume.

Before running Volume Preparation C, the user must decide whether he wants it to assign substitutes for bad tracks on the volume.

Track Substitution Not Requested

If the user does not wish Volume Preparation C to assign substitutes for bad tracks on the volume, he merely creates a job control file as instructed in Section II of this manual. If a bad track is encountered on the volume during formatting, a halt or console message occurs. In addition, the address of the bad track is given by the following message, which appears on the printer: CYL nnn TR nnn ERROR (nnn is a decimal value).

If desired, the user can achieve the track substitution capability at a later time by running Volume Modification D (see Section VII of this manual).

NOTE: When allocating files to a volume that does not contain *BADTRACKS and *VOLSPARES files, the user must be careful not to include in the units of allocation any bad tracks that have not been assigned substitutes.

Track Substitution Requested

When the user wishes Volume Preparation C to assign substitutes for bad tracks on the volume, he must include the appropriate special values in the job control file.[1]

1. The spares parameter <u>must</u> be included (with an appropriate value) to create the *BADTRACKS and *VOLSPARES files.
2. The FROM and TO parameters are used only when the *BADTRACKS and *VOLSPARES files are to be created in a nonstandard area. (In this case, the spares parameter must have the value SPARES = RELOCATE,.)
3. A Track statement <u>may</u> be used to declare a bad track, known to the user before the run, for which a substitute is requested. Track statements are especially useful when the entire volume is not to be surface checked by formatting (because substitutes will not be assigned for bad tracks in the unformatted area unless these bad tracks are declared in Track statements).

[1] See "Job Control Language Supplement," later in this appendix, for a complete description of job control language elements specific to bad track handling during Volume Preparation C.

When the *BADTRACKS and *VOLSPARES files have been created on the volume during Volume Preparation C, track substitution occurs automatically, as follows:

1. A substitute track is assigned for every bad track declared in a Track statement.
2. A substitute track is assigned for every bad track encountered in the area formatted during the run.[1]

No halt or console message occurs when a substitute is assigned for a bad track.

Just before end of run, the printer lists (1) each bad track declared in a Track statement, (2) each bad track encountered in the formatted area, and (3) all substitute tracks. This printer listing is called the list file.

Special Situations

SITUATION 1

A special situation exists if Volume Preparation C is being run without the track substitution capability (no spares parameter has been included in the job control file) and a bad track is encountered in one of the following areas:

1. The bootstrap track,
2. The volume label track, or
3. A track in the volume directory.

If any of these tracks is bad, a halt or console message occurs. If the volume label track or a track in the volume directory is bad, Volume Preparation C must be run again with the spares parameter included in the job control file to create the *BADTRACKS and *VOLSPARES files. A substitute track can then be assigned for the bad track when it is encountered during this run (or the bad track can be declared in a Track statement). If the bootstrap track is bad, Volume Preparation C can be run again, as described above.

If no substitute track were provided for a bad bootstrap track, the volume could not be used as a system residence volume (because the volume could not be bootstrapped).

If no substitute were provided for a bad volume label track or a bad track in the volume directory, the volume could not be used.

SITUATION 2

Track substitution is not possible for a bad track in the *BADTRACKS file; therefore, a

[1]Volume Preparation C formats and surface checks either the entire volume or only certain critical tracks. See the "Surface-Check Parameter" in Section II of this manual.

bad track in the *BADTRACKS file cannot be declared in a Track statement. If a bad track is encountered in the *BADTRACKS file during formatting of the volume, a halt or console message occurs.

Volume Preparation C must be run again to relocate the *BADTRACKS and *VOLSPARES files. The spares parameter must have the value SPARES = RELOCATE,.) A pair of FROM and TO parameters must follow the spares parameter.

SITUATION 3

Track substitution is not possible for any unusable track on the volume. "Unusable" means that the track surface is bad and not even one bad-track track-linking record can be read. (It is very unlikely that any track will ever be unusable.)

If any of the tracks mentioned in "Situation 1" is unusable, the result is the same as if the track were bad and no substitute track were provided (see the last two paragraphs of "Situation 1").

If a track in the *BADTRACKS file is unusable, the result is the same as if the track were bad (see "Situation 2").

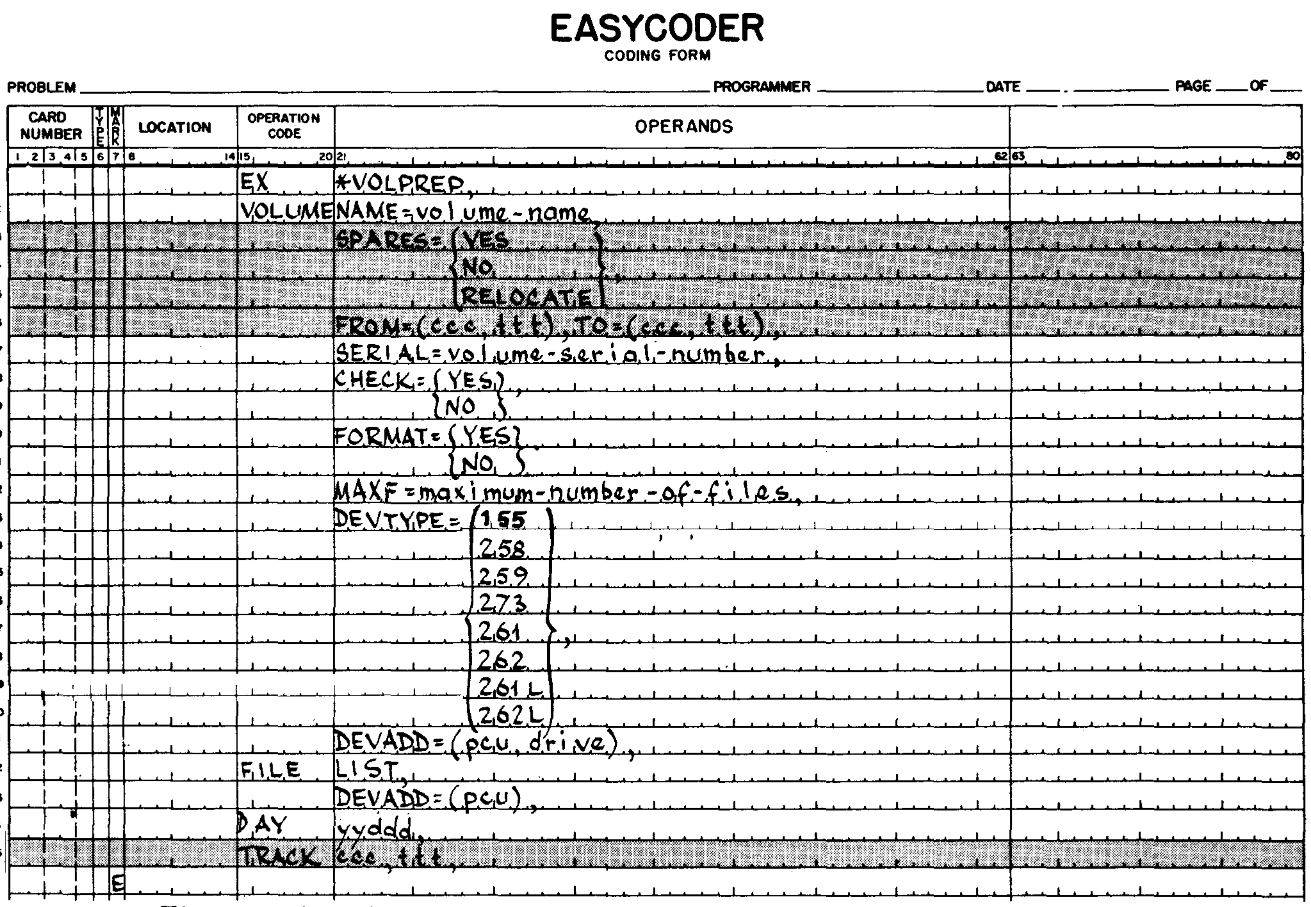
If any other track on the volume is unusable and *BADTRACKS and *VOLSPARES files have been created, the following message appears on the printer: CYL nnn TR nnn UNUSABLE (nnn is a decimal value).

Job Control Language Supplement

The following elements of job control language exist to assist the user in handling bad tracks during Volume Preparation C. These elements - the spares parameter, the FROM and TO parameters, and the Track statement - form a supplement to the job control language for Volume Preparation C described in Section II of this manual. The placement of these elements in the job control file is illustrated in Figure B-1.

SPARES PARAMETER

The spares parameter must be included in the job control file when the user desires to have Volume Preparation C create the *BADTRACKS and *VOLSPARES files. The spares parameter can appear anywhere in the Volume statement.



EASYCODER
CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

```
CARD    T M  LOCATION  OPERATION  OPERANDS
NUMBER  Y A            CODE
        P R
        E K
 1                     EX         *VOLPREP,
 2                     VOLUME     NAME=volume-name,
 3                                SPARES={YES     },
 4                                       {NO      }
 5                                       {RELOCATE}
 6                                FROM=(ccc,ttt),TO=(ccc,ttt),
 7                                SERIAL=volume-serial-number,
 8                                CHECK={YES},
 9                                      {NO }
10                                FORMAT={YES},
11                                       {NO }
12                                MAXF=maximum-number-of-files,
13                                DEVTYPE={155 },
14                                        {258 }
15                                        {259 }
16                                        {273 }
17                                        {261 }
18                                        {262 }
19                                        {261L}
20                                        {262L}
21                                DEVADD=(pcu,drive),
22                     FILE       LIST,
23                                DEVADD=(pcu),
24                     DAY        yyddd,
25                     TRACK      ccc,ttt,
26        E
```

Figure B-1. Placement of Supplementary Elements in Job Control File

If the user specifies the value SPARES = YES, the *BADTRACKS and *VOLSPARES files will be created in a standard area on the volume. If the user specifies the value SPARES = RELOCATE, the *BADTRACKS and *VOLSPARES files will be created in an area specified by the user in a pair of FROM and TO parameters. (When the RELOCATE option is specified, the spares parameter <u>must</u> be followed by a pair of FROM and TO parameters.)

The form of the spares parameter is shown below.

$$\boxed{\text{SPARES} = \left\{\begin{array}{l}\text{YES}\\ \text{NO}\\ \text{RELOCATE}\end{array}\right\},}$$

<u>YES</u> = A *BADTRACKS and a *VOLSPARES file are to be created in a standard area on the volume being prepared. The two files will occupy all of the tracks in the following standard areas.

Device Type	Cylinders
155	200 - 202
258	101 - 103
259, 273, 259A, 259B	200 - 202
261, 262	125 - 127

NO = The *BADTRACKS and *VOLSPARES files are not to be created.

RELOCATE = A *BADTRACKS file and a *VOLSPARES file are to be created in an area specified by the user. This option may be used if these files cannot be created in the standard area. The RELOCATE option must be followed immediately by a pair of FROM and TO parameters.

Default assumption: NO

FROM AND TO PARAMETERS

A pair of FROM and TO parameters is used only when the RELOCATE option has been specified in the spares parameter. The pair of FROM and TO parameters enables the user to designate the area on the volume where the *BADTRACKS and *VOLSPARES files are to be created. The pair of FROM and TO parameters must immediately follow the spares parameter in the job control file (as shown in Figure B-1).

The form of this parameter is shown below.

```
FROM = (ccc, ttt), TO = (ccc, ttt),
```

ccc = The cylinder address, in decimal, where leading zeros are not required.

ttt = The track address, in decimal, where leading zeros are not required. The first track on each cylinder is numbered zero.

The pair of FROM and TO parameters specifies that the *BADTRACKS and *VOLSPARES files are to occupy the following area of the mass storage volume: all the tracks from the track stated in the FROM parameter to the track stated in the TO parameter (inclusive) on each cylinder from the cylinder stated in the FROM parameter to the cylinder stated in the TO parameter (inclusive).

The area designated by a pair of FROM and TO parameters can be as large as the user desires or as small as two tracks (one track for the *BADTRACKS file and one for the *VOLSPARES file). The user-designated area can be located anywhere on the volume except where it would conflict with the bootstrap track, the substitute bootstrap track, the volume label, or the volume directory.

TRACK STATEMENT

A Track statement can be used to indicate the address of a bad track for which a substitute is to be made. When a Track statement is used, the address of the bad track must be known before Volume Preparation C is run.

The use of Track statements requires that the *BADTRACKS and *VOLSPARES files be created by Volume Preparation C.

A separate Track statement is required for each bad track that the user wishes to declare.[1] Any number of Track statements may be included, but they must follow all other job control statements of Volume Preparation C.

The format and placement of the Track statement are shown in Figure B-1. The cylinder and track address of each bad track (ccc, ttt,) is expressed in decimal numbers: leading zeros are not required.

Operator Communication and Control Supplement

The following halts and console messages are related to conditions involving bad or unusable tracks when Volume Preparation C is being run. The user should also note the pp2d halt (specific code 10) in Table 2-5 and the WRITE ERROR console message (specific code 8) in Table 2-8: both tables appear in Section II of this manual.

CONTROL PANEL HALTS

Table B-2. Supplementary List of Halts Specific to Volume Preparation C

B-Address Register Contents	Condition	Operator Action[1]
6502	There is a bad track in the *BADTRACKS file. (The *BADTRACKS file must be relocated.)	E = exit to the Supervisor.
6503	The volume label track or a track in the volume directory is unusable.[2] (The volume cannot be used.)	E = exit to the Supervisor.
6504	The bootstrap track (cylinder 000, track 000) is unusable.[2] (The volume cannot be used as a system residence volume; i.e., it cannot be bootstrapped.)	A = accept the condition and continue preparation of the volume. E = exit to the Supervisor.

[1] The user is not required to include a Track statement to declare a bad track if that track is in an area that is to be formatted by Volume Preparation C. When the bad track is encountered during formatting, a substitute track will be assigned for it (if the *BADTRACKS and *VOLSPARES files have been created).

Table B-2 (cont). Supplementary List of Halts Specific to Volume Preparation C

B-Address Register Contents	Condition	Operator Action[1]
6506	An attempt is being made to process a Track statement, but the *BADTRACKS and *VOLSPARES files have not been created on the volume.	E = exit to the Supervisor.

[1]The operator enters the appropriate response character and presses RUN ($A = 21_8$, $E = 25_8$).

[2]"Unusable" means that the track surface is bad and not even one bad-track track-linking record can be read. (It is very unlikely that any track will ever be unusable.)

CONSOLE MESSAGES

Table B-3. Supplementary List of Messages Specific to Volume Preparation C

Typewriter Message	Condition	Operator Action[1]
BAD TRACK IN *BADTRACKS FILE	There is a bad track in the *BADTRACKS file. (The *BADTRACKS file must be relocated.)	E = exit to the Supervisor.
UNUSABLE TRACK IN DIRECTORY OR LABEL	The volume label track or a track in the volume directory is unusable.[2] (The volume cannot be used.)	E = exit to the Supervisor.
UNUSABLE BOOT-STRAP TRACK	The bootstrap track (cylinder 000, track 000) is unusable.[2] (The volume cannot be used as a system residence volume; i.e., it cannot be bootstrapped.)	A = accept the condition and continue preparation of the volume. E = exit to the Supervisor.
*BADTRACKS NOT ON VOLUME	An attempt is being made to process a Track statement, but the *BADTRACKS and *VOLSPARES files have not been created on the volume.	E = exit to the Supervisor.

[1]The operator types the appropriate alphabetic response character and presses the space bar.

[2]"Unusable" means that the track surface is bad and not even one bad-track track-linking record can be read. (It is very unlikely that any track will ever be unusable.)

BAD TRACK HANDLING PROCEDURES FOR MASS STORAGE SORT C

The presence of bad tracks in any of the volume areas used for work files by Mass Storage Sort C slightly alters the calculations involved in determining whether to use single or double buffering.

Table B-4. Disk Table (Bad Tracks Present on Type 155, 258, 259, 273, 259A, or 259B Disk Pack Drives)

Column A	Column B
Data Characters Per Block	Records Per Block
241	1
741	3
1,241	5
3,741	15

Table B-5. Disk Table (Bad Tracks Present on Type 261 or 262 Disk Files)

Column A	Column B
Data Characters Per Block	Records Per Block
241	1
491	2
741	3
1,241	5
1,491	6
2,491	10
3,741	15
7,491	30

When the work files are on Type 155, 258, 259, 273, 259A, or 259B Disk Pack Drives, use Table B-4. When the work files are on Type 261 or 262 Disk Files, use Table B-5. Select from column A of the appropriate disk table the least value that is greater than or equal to the sort-item size (SIS). Let that value be represented by PBS (physical block size).

Using the PBS value determined from Table B-4 or B-5, refer to Appendix A to calculate whether single or double buffering will be used. (In Appendix A, Table A-1 or A-2 is used to determine the PBS, based on the assumption that there are no bad tracks present within the work files.)

BAD TRACK HANDLING PROCEDURES FOR DISK/TAPE COPY C

The following paragraphs describe Disk/Tape Copy C procedures when *BADTRACKS and *VOLSPARES files are present.

Disk-to-Tape Copy — Track Substitution on Input Disk

If the input mass storage volume contains *BADTRACKS and *VOLSPARES files and track substitution has occurred for one or more bad tracks in an area to be copied, the following action takes place (regardless of the copy method): when a bad input track is encountered, data is copied from the substitute track designated by the bad-track track-linking record (TLR) on the bad track. Header areas on the output tape record are modified to show the address of the bad track itself; thus, the output tape record is written as if no bad tracks existed on the input mass storage volume.

Tape-to-Disk Copy — Track Substitution on Output Disk

When *BADTRACKS and *VOLSPARES files exist on the output mass storage volume, the following action occurs: before data is copied to it, each output disk track is searched for a bad-track TLR. If none is found, the input data is copied to the output disk track. If, however, an output track contains a bad-track TLR, the input data is copied to the substitute track indicated by the bad-track TLR. (The bad-track TLR presumably links to a substitute track in the *VOLSPARES file.) No halt or console message occurs but a BAD TRACK TLR message appears on the printer.

PRECAUTION

When file data is copied to a substitute track on an output disk (because the intended output track was bad), the *BADTRACKS file must reflect the file-related status of the bad track. If the file-related status of a bad output track has not already been established by allocate or Volume Modification D, the user must follow one of the procedures outlined below.

*

*

Procedure A

1. Run Volume Preparation C or Volume Modification D for the output disk, including the spares parameter to create the *BADTRACKS and *VOLSPARES files. (If *BADTRACKS and *VOLSPARES files already exist on the output disk, this step can be omitted.)
2. Allocate to the output disk all files involved in the copy.[1] (The file-related status of each bad track is now established.)
3. Perform a tape-to-disk copy run, copying from the tape all desired areas except the volume directory and the *BADTRACKS and *VOLSPARES files.

Procedure B

1. Allocate to the output disk all files involved in the copy.[1]
2. Run Volume Modification D for the output disk, including (1) a spares parameter to create the *BADTRACKS and *VOLSPARES files and (2) a Track statement for each bad track. (The file-related status of each bad track is now established.)
3. Perform a tape-to-disk copy run, copying from the tape all desired areas except the volume directory and the *BADTRACKS and *VOLSPARES files.

Disk-to-Disk Copy

TRACK SUBSTITUTION ON INPUT DISK

The procedure is the same as that described under "Disk-to-Tape Copy" except that the output volume is mass storage.

TRACK SUBSTITUTION ON OUTPUT DISK

The procedure is the same as that described under "Tape-to-Disk Copy" except that the input volume is mass storage.

Suggested Method for Creating Disk Backup With Track Substitution Capability

The following procedure is suggested when the user desires to create disk backup with track substitution capability.

[1] See also "Precautions When Using Disk/Tape Copy C" in Section VI if the copy-by-rectangle method is to be used.

PHASE 1 — CREATING A BACKUP TAPE

Perform a disk-to-tape copy run using the copy-by-rectangle method. (The input disk is the volume for which backup is desired.) Specify two (or more) pairs of FROM and TO parameters in the Input Volume statement: the first pair must delimit only the input volume directory, the second (and any subsequent) pair must delimit only the input disk area for which backup is desired. (No portion of the *BADTRACKS and *VOLSPARES files is to be included in the area delimited by the pairs of FROM and TO parameters.)

PHASE 2 — COPYING FROM THE BACKUP TAPE TO AN OUTPUT DISK

1. Perform steps 1 through 3 outlined for "Procedure A" under "Tape-to-Disk Copy."
2. Perform a tape-to-disk copy run, copying to the output disk the second (and any subsequent) rectangle copied to the backup tape in phase 1, above. (The volume directory of the original input disk has been copied to the output disk by step 2 of Procedure A.)

COMPUTER GENERATED INDEX

Honeywell

Cut Along Line

HONEYWELL
TECHNICAL PUBLICATIONS REMARKS FORM *

TITLE:	MOD 1 (MSR) UTILITY ROUTINES	DATED:	NOVEMBER, 1968
		FILE NO:	123.6905.141C.3-619

ERRORS NOTED IN PUBLICATION:

Fold

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

Fold

(Please Print)

FROM: NAME ______________________ DATE ____________

COMPANY ______________________

TITLE ______________________

ADDRESS ______________________

____

* Your comments will be promptly investigated by appropriate technical personnel, action will be taken as required, and you will receive a written reply. If you do not require a written reply, please check here ☐.

FIRST CLASS
PERMIT NO. 39531
NEWTON HIGHLANDS
MASS.

BUSINESS REPLY MAIL
No postage stamp necessary if mailed in the United States
POSTAGE WILL BE PAID BY

HONEYWELL
151 NEEDHAM STREET
NEWTON HIGHLANDS, MASS. 02161

ATT'N: MARKETING INFORMATION SERVICES, MS 251

Cut Along Line

Honeywell