HONEYWELL

SOFTWARE

MOD 1 (MSR)

SUPERVISOR

GENERAL SYSTEM: SERIES 200/OPERATING SYSTEM - MOD 1 (MASS STORAGE RESIDENT)

SUBJECT: Description of the Supervisor, the General Control Program for the Mass Storage Operating System.

SPECIAL INSTRUCTIONS: This edition completely supersedes the manual of the same name dated January 16, 1967, Addendum No. 1 dated May 5, 1967, and Addendum No. 2 dated August 25, 1967. New and changed information in this edition relates primarily to the incorporation of a multiprogramming capability into the Mod 1 (MSR) system. This manual is one of a series describing the Mass Storage Operating System. Refer to the Preface for other related documentation.

INCLUDES UPDATING PAGES PUBLISHED AS ADDENDUM NO. 1 ON DECEMBER 11, 1967, AS ADDENDUM NO. 2 ON JANUARY 29, 1968, AS ADDENDUM NO. 3 ON AUGUST 16, 1968, AS ADDENDUM NO. 4 ON OCTOBER 31, 1968, AS ADDENDUM NO. 5 ON AUGUST 6, 1969, AND AS ADDENDUM NO. 6 ON DECEMBER 15, 1969.

DATE: October 31, 1967

FILE NO.: 123.5005.141C.1-616*

0669
15.5M
7.5170
Printed in U.S.A.

*Underscoring denotes Order Number.

PREFACE

This manual describes the Supervisor, which is the general control program for the Series 200/Operating System — Mod 1 (Mass Storage Resident). It is one of a series of manuals that describe the entire operating system. Besides this manual, other pertinent publications include the following:

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

Program Development Subsystem (Order No. 617);

Data Management Subsystem (Order No. 618);

Utility Routines (Order No. 619); and

Operating Procedures (Order No. 620).

The introductory bulletin cited above is prerequisite reading to this manual and the other manuals listed. In addition, a publications guide is provided in the introductory bulleting to aid the reader in his study of the system.

Section I of this manual describes the basic elements and equipment requirements of the Supervisor. Section II gives the reader the necessary information to use the Supervisor to perform basic functions, and Section III provides programming considerations. Section IV outlines the organization of the Supervisor; Section V gives, in detail, the Supervisor functions; and Section VI presents the method of creating the Supervisor. Section VII describes operation in the foreground/background environment; Section VIII contains operating procedures; and Appendix A defines the communication area. To perform the functions learned in Sections II and III, the reader may bypass Sections IV through VII and go directly to Section VIII. Sections IV through VII are specifically for the programmer who plans to use other than the basic functions.

The Mod 1 (MSR) Supervisor is a coded program designed to extend the power of Series 200 in the area of operations control. It is supported by comprehensive documentation and training; periodic program maintenance and, where feasible, improvements are furnished for the current version of the program, provided it is not modified by the user.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

All operations in the Mass Storage Operating System are performed under the general control of the Supervisor. The Supervisor performs two main functions: (1) job control, and (2) loading and starting program segments.

JOB CONTROL

Job control denotes the process of automatically sequencing from one job to the next. The Supervisor performs this function based on information read from the job control file. The job control file is the input file of control information identifying a job and defining its requirements. The Supervisor reads a statement from the job control file and then activates the appropriate element of the system to perform the job. When the job is completed, the activated element informs the Supervisor to read from the job control file again. This sequence of events continues until the input in the job control file is exhausted.

PROGRAM LOADING

Program loading consists of locating the appropriate system program or user program in the program file on the mass storage volume, loading it into main memory, and starting its execution. The Supervisor determines which program to load by reading the job control file. Several options are provided to control the searching, loading, and starting sequence so that the programmer has complete freedom to set up exactly the desired sequence of functions.

FOREGROUND/BACKGROUND ENVIRONMENT

The Supervisor can control the simultaneous execution of two programs in a manner called foreground/background operation. The Supervisor controls the sharing of central processor cycles between the two programs by allowing one to continue operation while the other is waiting for the completion of an input/output operation. This multiprogramming capability is an integral part of the Supervisor. It is fully described in Section VII of this manual and pertinent information on foreground/background operation is also included in Section VIII.

EQUIPMENT REQUIREMENTS

Basic Equipment Requirements

The following equipment is required by the Supervisor when it is operating in the single-job environment; additional equipment is required when the Supervisor is operating in the foreground/background environment.

1. A Series 200 central processor with at least 12,288 characters of main memory (see Table 7-1) and the specific capabilities required by the system (see the Programs Library Catalog, Order No. A52, for details). The Supervisor requires 1400 main memory locations for its smallest version (3-character address mode without console typewriter). In addition, 130 locations are used for the communication area (locations 0 and 61-189).
2. Advanced Programming Instructions.
3. One direct-access device and associated control. The possible combinations of devices and controls are tabulated below:

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

4. One card reader.
5. Index registers X5 and X6.

Additional Usable Equipment

Additional core storage (and/or equipment) is required to use certain optional features of the Supervisor. Table 1-1 shows the total number of locations required for each of these features. (The communication area is not included.) These values are approximate and subject to change.

Table 1-1. Core Storage Required by Supervisor

Version of Supervisor	3-character Address Mode		4-character Address Mode	
	Basic	Program Test	Basic	Program Test
Basic (control panel)	1400	5400	2000	7100
Console Typewriter	2000	6000	2600	7700
Multiprogramming Control	4200		5400	

LARGER CORE STORAGE

The Supervisor can control programs using up to 262,144 locations of core storage. It must be created in 4-character address mode when controlling programs which use more than 32,768 locations.

CONSOLE TYPEWRITER

Communication with the operator can be through a Type 220-1, -2, or -3 Console.

PROGRAM TEST FUNCTION

When Program Test C is run, the Supervisor requires an additional 4000 characters of main memory for 3-character address mode and an additional 5100 characters of main memory for 4-character address mode. Seven additional disk devices are optional when running Program Test C. For additional information about the Program Test function, see the Program Development Subsystem manual, Order No. 617.

MULTIPROGRAMMING CONTROL

The following equipment is required for use of the multiprogramming control version of the Supervisor.

1. Program Interrupt, (Feature 012), if the central processor is a Type 201.
2. 16, 384 main memory locations.
3. One Type 220-1, -2, or -3 console.
4. INTERRUPT button, on either the control panel or the console.
5. At least one peripheral control with the peripheral interrupt capability (used by the foreground program).

Table 7-1 lists the specific combinations of central processor types, optional instruction features, and consoles that meet the requirements for foreground/background operation.

SECTION II

BASIC DESCRIPTION OF THE SUPERVISOR

This section describes the basic functions of executing a job, loading a program segment, and exiting to the Supervisor when a job is completed. If these are the only functions of interest, the reader may bypass Sections IV through VII. Note that this section applies only to the single-job environment; information on foreground/background environment (multiprogramming) is contained in Section VII.

FUNCTIONS OF THE SUPERVISOR

The Supervisor is bootstrapped initially into main memory from mass storage. This operation is performed once; the Supervisor can then function continuously without being reloaded.

Once in memory, the Supervisor is ready to perform its functions of job control and program segment loading. The Supervisor is controlled by the user in either of two ways: (1) job and program sequencing are controlled by job control statements, or (2) segment loading within a program is controlled by programmed calls to the Supervisor.

Executing a Job or Program

The Supervisor starts by reading the job control file (which is on punched cards). The Supervisor searches for an Execute statement; when encountered, the Execute statement directs the Supervisor to locate, load, and start a named program segment. Alternatively, the operator may enter the Execute statement through the control panel.

Program Segment Loading

When the current segment of the program has completed execution, it transfers control to the Supervisor to load another segment. To do this, the program executes a call to the Supervisor. The Supervisor then searches the directory of the machine-language program file for the mass storage address of the specified segment, loads this segment into the locations assigned by assembly or compilation, and starts execution of the segment.

Normal Exit from a Program

When a program has completed operation and does not require execution of another segment, it exits to the Supervisor. The Supervisor then reads the job control file for an Execute statement and starts the new job or program.

COMMUNICATING WITH THE SUPERVISOR

The following paragraphs describe the two methods by which the user communicates with the Supervisor: (1) job and program sequence control, by means of job control statements, and (2) program segment loading and exiting, by means of programmed calls to the Supervisor.

The programmer uses program instructions to communicate with the Supervisor. The exact instructions used for program segment loading and exiting depend on the current address mode of the program issuing the call.

The Execute Statement

The Execute statement directs the Supervisor to locate, load, and start a named program segment. The format of the Execute statement is shown in Figure 2-1.

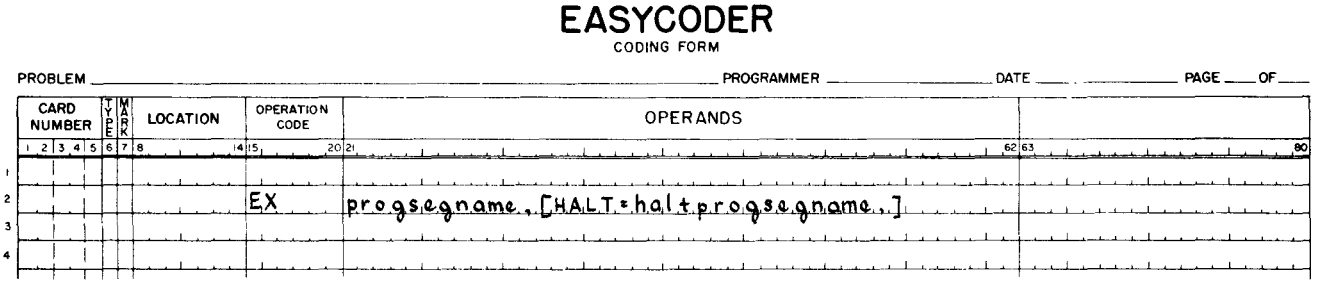

Figure 2-1. Execute Statement

Description:

progsegname	—	The program and segment name of the program segment to be executed.
haltprogsegname	—	The Supervisor halts after the named program segment is loaded. The brackets [ ] indicate that this parameter is optional.

A program segment name consists of eight characters; the first six are the program name, and the last two are the segment name. The characters must be chosen from the letters A through Z and the digits 0 through 9. Trailing spaces may appear either in the program name (first six characters) or in the segment name (last two characters), or in both. No other spaces and only the characters listed above can appear in a program segment name. The program segment name must appear in character locations 21-28; the comma must appear in location 29. The halt-program-segment-name parameter and its associated comma, if specified, can appear in any 14 contiguous locations between locations 30 and 80.

Loading a Program Segment

When a program is ready to load its next segment, it executes a call to the Supervisor. This call sets parameters in the communication area and branches to the Supervisor. The symbolic coding for loading a program segment in 3-character address mode is shown in Figure 2-2; the symbolic coding for loading a program segment in 4-character address mode is shown in Figure 2-3. The MCW instruction addresses a constant (DCW) that contains the 2-character name of the segment to be loaded. Thus, the name is moved to the segment name field of the Supervisor; the Branch instruction to the Supervisor is then executed.

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
				MCW	sname,75
				B	13Ø
			sname	DCW	#2Asg

sname=TAG OF CONSTANT CONTAINING SEGMENT NAME.
sg = TWO-CHARACTER SEGMENT NAME.

Figure 2-2. Loading a Program Segment in 3-Character Address Mode

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
				MCW	sname,75
				B	(168)
			sname	DCW	#2Asg

sname = TAG OF CONSTANT CONTAINING SEGMENT NAME.
sg = TWO-CHARACTER SEGMENT NAME.

Figure 2-3. Loading a Program Segment in 4-Character Address Mode

Normal Exit from a Program

When a program has completed operation and does not require execution of another segment, it exits to the Supervisor by means of an indirect branch. The symbolic coding for the normal exit from a program in 3-character address mode is shown in Figure 2-4; the symbolic coding for the normal exit from a program in 4-character address mode is shown in Figure 2-5.

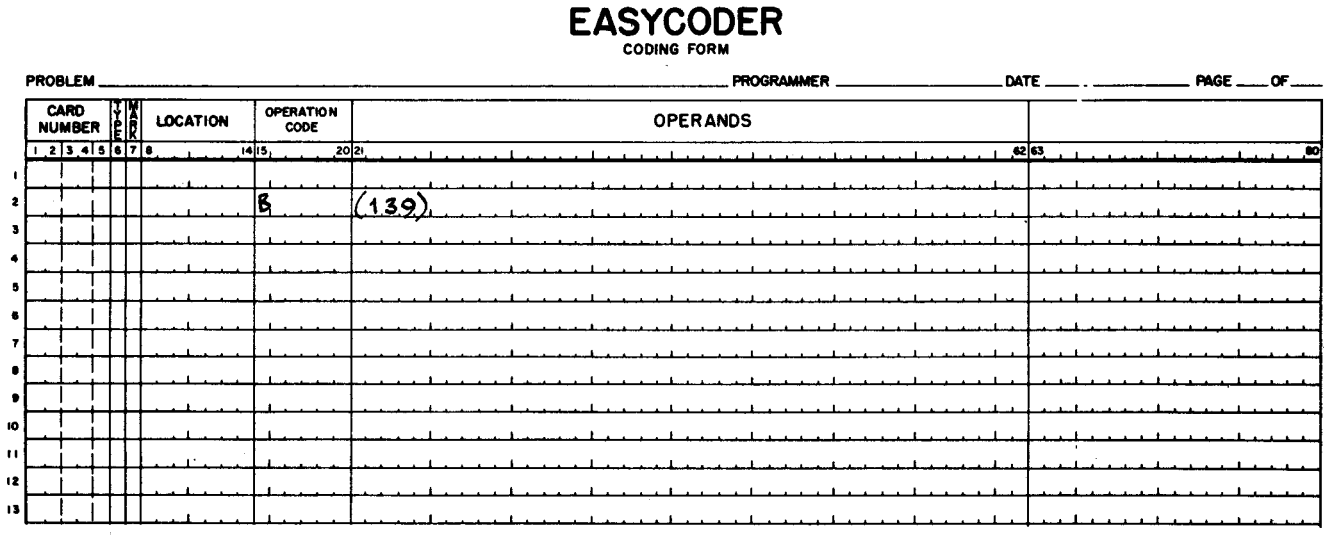

Figure 2-4. Normal Exit from a Program in 3-Character Address Mode

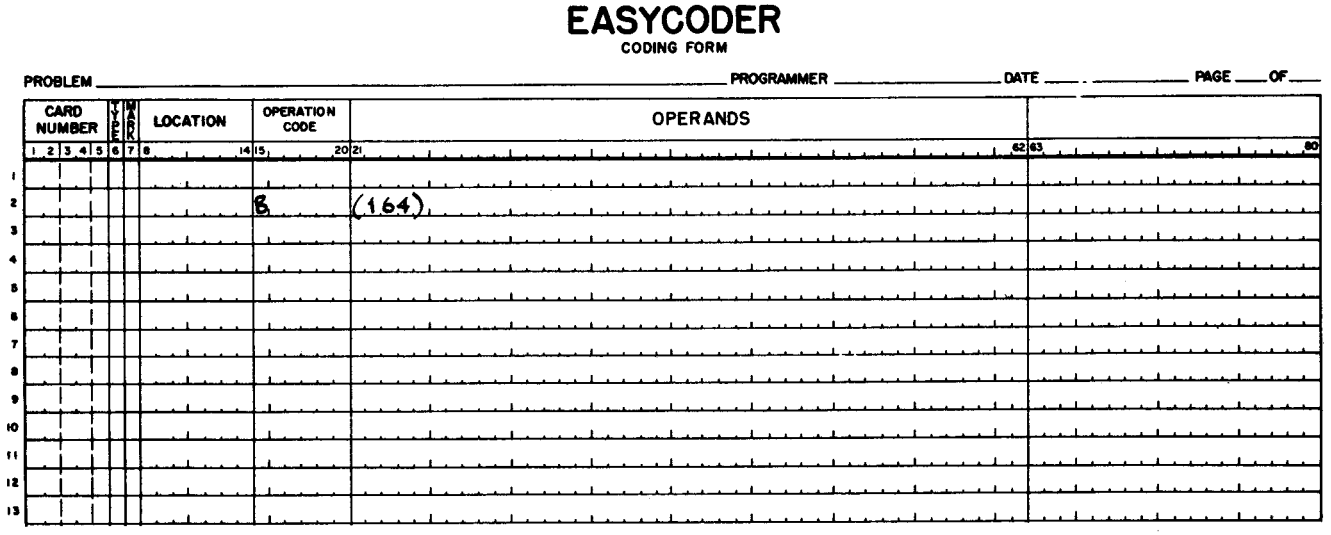

Figure 2-5. Normal Exit from a Program in 4-Character Address Mode

SECTION III

PROGRAMMING CONSIDERATIONS

This section contains pertinent considerations which the programmer should be aware of before writing mass storage programs. It does not contain programming considerations for operation in the foreground/background environment. These latter considerations will be found in Section VII. All storage locations are given in decimal form unless otherwise noted.

1. Since the Supervisor resides in high memory and has a variable starting location, care must be exercised to ensure that no program overlaps the Supervisor. In particular, when operating programs with a variable amount of memory, the address stored in the highest memory-location-available field (locations 187-189) must be considered when computing the memory available.

2. A Supervisor created in 3-character address mode can only load programs into the first 32,768 locations of main memory and always starts program segments in 3-character address mode.

3. A Supervisor created in 4-character address mode can load into the first 262,144 characters. It will start a program or a segment in an address mode dependent on the location to which the starting branch is made. If this address is greater than 32,767, the branch is executed in 4-character address mode; otherwise, it is executed in 3-character address mode.

4. The Supervisor uses and does not restore index registers X5 and X6. These registers have word marks at their high-order locations after loading; but the user is cautioned that the address mode used by the Supervisor does not necessarily correspond to that used by the loaded program segment, and the word marks may not be where the loaded program segment requires them. Index register X6 contains an address one higher than that at which the last character of the called segment was loaded. Index register X5 contains the address of the control character in the buffer that terminated the loading operation. The three characters at the locations immediately following the address specified by the index register X5 will contain the normal starting address of the segment just loaded.

5. Own-code routines must not destroy the contents of index registers X5 and X6.

6. The Supervisor does not use or disturb any locations below location 61, with the exception of index registers X5 and X6 and location 0.

7. If a program uses the Change Address Mode (CAM) instruction to alter the trapping mode, then the program should also make the corresponding change in the trapping mode field (location 147).

8. If the Supervisor has been conditioned to provide program test facilities, own-coding is not permitted.

9. A program should terminate in one of two ways.

   a. Normally: The program exits to the Supervisor via a B(139) or a B(164).

   b. Emergency: Either the program branches to the emergency exit location (B 86), or the operator terminates the program manually and sets the sequence register to 126(Octal).

SECTION IV
ORGANIZATION OF THE SUPERVISOR

STRUCTURE OF THE SUPERVISOR

The Supervisor is a segmented program which occupies two areas in main memory — the communication area and the floating area. All storage locations are given in decimal form unless otherwise noted.

Communication Area

The communication area is a fixed area in lower memory that includes location 0 and locations 61 through 189. The index registers, locations 1 through 60, are available to the user. However, the Supervisor also uses and does not restore index registers X5 and X6. The communication area contains the information necessary for the Supervisor to perform its functions of searching, loading, and starting. The contents of the communication area are outlined in Appendix A.

Floating Area

The floating area is an area in upper memory whose size depends on the selection of Supervisor features made at system generation time. This portion of the Supervisor can be "floated" to the upper memory location in two ways: (1) at execution time, when the Supervisor can relocate itself to the highest bank (unit of 4,096 characters) of memory, or (2) at system generation time, when the Supervisor can be specialized and assembled to reside permanently in the highest bank of memory.

The advantage of floating a portion of the Supervisor is that it is possible to provide a common origin for all programs that operate within the system. The highest address used by the communication area is location 189. Programs must be assembled above this area. If the remainder of the Supervisor were to be placed immediately behind the communication area, the origin of operated programs would vary because the size of the Supervisor varies.

Although the floating portion of the Supervisor does vary in size, a field in the communication area (locations 187 to 189, see Appendix A) always contains the highest memory location available to operating programs. Thus, the user always knows the lowest and highest memory addresses available to operating programs.

Part of the floating area contains resident routines that are always in memory. A second part of the floating area is an overlay area where less frequently required routines of the Supervisor are brought in as they are needed.

EXECUTABLE PROGRAM FILE

Programs to be operated under control of the Supervisor are stored in the permanent executable program file (residence file) on mass storage. Loading is normally from the residence file; but, under Program Test, it may also be from the go file. An executable program file is a partitioned sequential file composed of two areas: a directory area and a program data area containing the program segments.

Program Directory

The directory is defined as the member index of the file giving the name and mass storage location for every program segment in the file. Each item in the directory refers to one program segment.

The capacity of the directory (which determines the number of program segments that may be contained in the file) is specified by the user when the executable program file is created through a file-support allocation run.

Program Segments

The data area of the executable program file contains the program segments. One program segment is located and loaded as the result of a programmed call to the Supervisor. The area allocated by the user to the data portion of the executable program file determines the total amount of code that can be stored in the file.

SECTION V
DETAILED DESCRIPTION OF SUPERVISOR FUNCTIONS

INTRODUCTION

This section provides a detailed description of the functions the Supervisor can perform. It is intended for the programmer who wishes to use the Supervisor for functions other than those described in Section II. The programmer performs these additional functions by writing instructions to alter the communication area and to branch to the Supervisor. The programmer thus supplies to the Supervisor the parameters necessary for performing the desired functions. This section contains information for single jobs; multiprogramming information is found in Section VII.

The format of the communication area is shown in Table A-1 of Appendix A, and the functions and necessary parameters are summarized in Table A-2.

The following paragraphs define the Supervisor functions and specify the parameters used (location, initial value, possible values, reset conditions). All storage locations are given in decimal form unless otherwise noted.

SEARCHING

The following paragraphs define the parameters related to the process of searching for the desired program segment.

Normally, the Supervisor searches the executable program file directory for the mass storage address of the called program segment. The record at this address is then read and checked to see if it is a segment header record. If there is no directory entry for the called program segment or if the first record is not a segment header record, the Supervisor halts.

If the Supervisor was directed to load from a specified mass storage address (i.e., search mode 07), the directory search routine is not executed.

Search Mode

The search mode field (location 111) indicates the type of search to be performed, as shown by the parameters in the "contents" column in Table 5-1.

Table 5-1. Search Mode Parameters

Contents	Function
20	Search for and load the program segment with the specified program and segment name.
22	Search the executable program file directory for the specified program and segment name and supply the calling program with the mass storage address of the called segment. This address is conveyed through the program name field (locations 68-73 of the communications area) in the format CCTTRR (cylinder, track, record, in binary).
60	Search for and load the program segment with the specified program name, segment name, and visibility.
62	Search the directory for the specified program name, segment name, and visibility, and supply the calling program with the mass storage address of the called program segment (as in search mode 22).
07	Load the program segment at the mass storage address specified in locations 68 through 73 of the communication area. This address has the same format shown above under search mode 22. Search mode 07 cannot be used unless search mode 22 or 62 had been used previously to supply the calling program with the mass storage address of the called program segment.
01	Decimally add a value of 1 to the segment name specified in locations 74 and 75; then proceed as described for search mode 20.
NOTE: The action performed by the Mod 1 (MSR) Supervisor for this search mode differs from that performed by the Floating Tape Loader-Monitor C.	

The initial value of the search mode field is 20. It is reset to 20 after a normal end-of-job exit.

NOTE: Two search modes used by Floating Tape Loader-Monitor C of the Series 200/Operating System-Mod 1 (Tape Resident) are not applicable to the Supervisor. They are:

00 — Search for and load the segment with the specified segment name within the current program.

40 — Search for and load the segment with the specified segment name and visibility within the current program. *

The Supervisor operates on these codes as follows (see Table 5-1):

00 — Interpreted as search mode 20.

40 — Interpreted as search mode 60. *

Program Name

The program name field (locations 68-73) contains the program name to be used as a search key in search modes 20, 22, 60, and 62. The program name of the called program segment is inserted into these locations by the Execute statement in the job control file or by a call to the Supervisor. When a program segment is loaded, its program name is placed in this field by the Supervisor. If the search mode is 22 or 62 (so that no loading occurs after locating the program segment), the Supervisor places the mass storage address of the first record of the requested program segment into the program name field.

Segment Name

The segment name field (locations 74-75) contains the segment name to be used as a search key in search modes 20, 22, 60, and 62. The segment name of the called program segment is inserted in these locations by an Execute statement in the job control file or by a call to the Supervisor. When a program segment is loaded, its segment name is placed in this field by the Supervisor.

Visibility Mask

The visibility mask field (locations 113-118) contains the visibility mask to be used in search modes 60 or 62. A visibility match is obtained if there is at least one bit position containing a "1" in both the visibility mask and the visibility key of the requested program segment. The initial value is visibility A (40 00 00 00 00 00, in octal).

LOADING

If the search is successful, the Supervisor proceeds to load the called program segment into memory. Loading consists of reading and distributing each successive record of the program segment. Each record is read into a buffer. From there, the instructions and constants are distributed to specific memory locations and punctuated as specified by control characters in the record.

Between the reading and distributing phases of loading, it is possible to execute own-coding routines. After reading a record into the buffer, the Supervisor always branches to the own-code location. All own-code routines may then do one of two things: (1) return to use the Supervisor's own distribution routine (own-code return before distribution), or (2) perform its own distribution and return to the read routine of the Supervisor (own-code return after distribution).

A program segment may be loaded into an area higher than that for which it was translated (assembled or compiled) by using the relocation augment. However, the Supervisor does not perform address adjustment; the program segment is loaded into the new area exactly as it was translated.

At loading time, the program and segment names of a program segment to be loaded are placed in locations 68 through 75. After a program segment has been loaded, the Supervisor resets the relocation augment to 0 and the own-code exit to assume no own-coding.

Relocation Augment

The relocation augment field (locations 107-109) contains a value to be added to the address at which all the code of the called program segment is to be loaded. This augment is applied to instructions, constants, the addresses of areas to be cleared, and the normal start location of the program segment. Note that the code itself is not altered; the program segment is merely loaded into a new area. The initial value is 0. It is reset to 0 both after a program segment has been loaded and after a program exits.

Halt Name

The halt name field (locations 77-84) provides space for a program name (locations 77-82) and segment name (locations 83-84). After the program segment with this name has been loaded, the Supervisor halts. When the RUN button is pressed, the Supervisor continues as directed by the starting parameters.

The halt name field is a single field. The only word mark is at location 77. The halt name field may also be set by the "haltprogsegname" parameter of the Execute statement as described in Section II.

Exit to Own-Coding

A calling program may execute own-coding during the loading of a called program segment by setting up an appropriate branch in the communication area. The starting address of the own-code routine must be placed in locations 103 through 105. The own-code routine must be assembled in 3-character address mode and reside below location 32,768. This field (locations 103-105) is the A-address of a Branch instruction. No punctuation is present at these locations, and no punctuation may be placed there by calling programs. The branch is made immediately after each record is read. Before the branch, the Supervisor sets index register X5 to the address in main memory of the first character in the record. This exit is initially set to assume that there is no own-coding. It is reset to this same value whenever a program completes execution and exits or whenever a program segment is loaded.

Own-code Return Points

The own-code routine must return to the Supervisor with a branch to one of the two own-code return points in the communication area. These branches must be made in 3-character address mode.

OWN-CODE RETURN BEFORE DISTRIBUTION

If the return is made to location 122, the Supervisor performs record distribution in the normal way. When this return is used, the settings of X5 and X6 must not be altered by the own-code routine.

OWN-CODE RETURN AFTER DISTRIBUTION

If the return is made to location 126, the Supervisor bypasses normal record distribution and reads the next record. When this return is used, the own-code routine must recognize the last record of the called program segment. When this last record is recognized, return must not be made to location 126; instead, X5 must be set to the address of a location containing the character 61 followed by the three-character starting address of the program segment just loaded. Return is then made to location 122.

STARTING

After loading a called program segment, the Supervisor may return to the calling program segment or may branch to a normal or special starting location in the called program segment. If the Supervisor is specialized for 3-character address mode, the branch is always made in 3-character address mode. If the Supervisor is specialized for 4-character address mode, the starting address mode depends upon the address to which the starting branch is made. If this address is greater than 32,767, the branch is executed in 4-character address mode; otherwise, it is executed in 3-character address mode. Before executing the branch, the Supervisor uses the rightmost four bits of the trapping mode indicator (location 147) to set up and execute a Change Addressing Mode (CAM) instruction that sets the trapping mode indicator of the central processor.

Starting Mode

The starting mode field (location 112) specifies one of the three alternative locations to which the Supervisor transfers control after loading. It must contain one of the following three values:

- N — Branch to the location specified as the normal starting location in the called program segment. The relocation augment is added before the branch is made.
- S — Branch to the address given by the special start field. The relocation augment is not added to this address.
- R — Branch to the location immediately following the one from which the call to the loader was made. The relocation augment is not added to the address.

The initial value of the starting mode field is N. It is reset to N when a program exits.

Special Start

The special start field (locations 119-121), which is used only with start mode "S" specifies the address to which control is to be transferred by the Supervisor after loading. This address may be any memory location up to 32,767 for a 3-character Supervisor and up to 262,143 for a 4-character Supervisor. The initial value of this field is 0; it is never reset by the Supervisor.

Trapping Mode

The trapping mode field (location 147) contains a character whose low-order four bits are substituted for the variant character of the Change Addressing Mode (CAM) instruction. This instruction is executed immediately before the Supervisor starts the called program to establish the trapping mode that will be in effect when the called program segment is started. The trapping mode field should contain one of the following two values (octal):

1. 00 — No item mark trapping or
2. 04 — Item mark trapping.

The initial value is 00.

If the trapping mode is specified, the operation code of any instruction which contains an item or record mark is both extracted and executed as if it were a Change Sequencing Mode (CSM) instruction, regardless of the operation code present. The CAM and CSM instructions and the trapping mode are described in detail in the Series 200 Programmers' Reference Manual (Models 200/1200/2200/4200), Order No. 139. This facility provides for automatic changes in program sequence without executing programmed instructions to initiate such changes.

The programmer is urged not to use the trapping mode in his programs because the program test facility uses the item-mark trapping feature to initiate memory dumps. A program using item-mark trapping will not be able to use certain dump facilities.

PROGRAM TERMINATION

There are two forms of program termination: normal exit and emergency exit.

Normal Exit

Two program exit location fields are provided in the communication area for a normal exit (also called Program Exit or General Return Address). A program running in 3-character address mode branches indirectly to location 139 when it has completed operation and does not require execution of another segment. A program running in 4-character address mode exits by executing an indirect branch to location 164. Execution of a program exit sets the Supervisor to perform the next job. The programmer provides for a normal exit by specifying B(139) or B(164) as the final instruction for each job.

Emergency Exit

One field (locations 86-89) is provided for an emergency exit. A program may branch to location 86 when it has encountered a condition it cannot correct.

If a program is running under control of Program Test C, an emergency exit initiates a memory dump of all memory locations up to and including the bank containing the Supervisor; it also results in terminal mass storage dumps and termination of the test. Control then passes to the next program development function.

If a program is not running under Program Test C, there are no dumps and control is returned to the Supervisor for the next job.

The branch to the emergency exit location must be executed in 3-character address mode. Thus, if a program is running in 4-character address mode or if it is running in 3-character address mode but is above location 32,767, the sector bits of the A-address register must be set to zeros before the program can change to 3-character address mode and execute the branch.

BOOTSTRAP INFORMATION

Relocation Bank Indicator

The Supervisor preserves the relocation bank indicator in location 62. Table 8-2 shows the possible values of the relocation bank indicator. The indicator, used when the Supervisor is bootstrapped into memory, shows the bank in which the Supervisor resides.

Bootstrap Device Identification

Location 63 of the communication area contains the address assignment of the peripheral control unit from which the Supervisor was loaded.

Supervisor Identification

Location 85 of the communication area contains the unique identification character of the Supervisor. (See parameter X in Section VI.)

OTHER FEATURES OF THE SUPERVISOR

Revision Number

Before starting to load a program segment, the Supervisor moves the program's revision number into the revision number field (locations 65-67). Other programs or the operator may reference this field.

Current Date

The operator may enter the current date into the current date field (locations 142-146) for reference by other programs. Locations 142 and 143 specify the year (00 to 99), and locations 144-146 specify the day of the year (001 to 366). The initial value of this field is 00000.

Highest Memory Location Available

The highest-memory-location available field (locations 187-189) contains the address of the highest memory location that may be used by any program. The floating portion of the Supervisor resides above this address. Any program computing the amount of memory available must take this value into account.

EXAMPLES OF PROGRAM CALLS TO THE SUPERVISOR

Example 1 - Loading a Specified Program Segment in 3-Character Address Mode

Call the program segment named PROCES AA (see Figure 5-1 below) and start it at its normal starting location. Note that the coding does not include entries for loading device, search mode, or start mode, since the desired values for these parameters are the initial values established by the Supervisor.

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					MCW	SGNAME,75	
3					MCW		
4					B	130	
5				PRNAME	DCW	#6APROCES	
6				SGNAME	DCW	#2AAA	
7							
8							
9							
10							

Figure 5-1. Example of Loading a Specified Program Segment in 3-Character Address Mode

Example 2 — Loading a Specified Segment by Visibility in 3-Character Address Mode

Call the program segment named INITPR NN (see Figure 5-2 below), also identified by visibility C or D, and start the specified segment at its normal starting location.

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
				MCW	SG,75
				MCW	
				MCW	SRCH,111
				MCW	VISIB,118
				B	13Ø
			PR	DCW	#6AINITPR
			SG	DCW	#2ANN
			SRCH	DCW	#1C6Ø
			VISIB	DCW	#6C14ØØØØØØØØØØ

Figure 5-2. Example of Loading a Specified Segment by Visibility in 3-character Address Mode

Example 3 — Return Start in 4-Character Address Mode

Call the program segment named PROCES AA (see Figure 5-3 below). Do not start execution after loading PROCES AA, but return to the next instruction in the program that made the call (the instruction which immediately follows B (168).

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
				MCW	SGNAME,75
				MCW	
				MCW	NOSTRT,112
				B	(168)
			PRNAME	DCW	#6APROCES
			SGNAME	DCW	#2AAA
			NOSTRT	DCW	#1AR

Figure 5-3. Example of Return Start in 4-Character Address Mode

Example 4 — Relocation and Special Start in 4-Character Address Mode

Call the program segment named AAAMEM S1 (see Figure 5-4 below), relocate the program segment 2,500 octal locations higher, and start AAAMEM S1 at octal location 2,510.

EASYCODER
CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____

	CARD NUMBER	TYPE	MOD	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1							
2					MCW	SN,75	
3					MCW		
4					MCW	RELOC,109	
5					MCW	STMODE,112	
6					MCW	SPST,121	
7					B	(168)	
8							
9				PN	DCW	#6AAAAMEM	
10				SN	DCW	#2AS1	
11				STMODE	DCW	#1AS	
12				RELOC	DCW	#3C00250Ø	
13				SPST	DCW	#3C00251Ø	
14							
15							
16							
17							

Figure 5-4. Example of Relocation and Special Start in 4-Character Address Mode

SECTION VI

CREATING THE SUPERVISOR

To generate a version of the Mass Storage Supervisor, it is necessary to perform an Easycoder assembly which specializes the routine @MSS from the macro routine library. (Easycoder assembly is described in the Program Development Subsystem manual. Information on generating the bootstrap routine is contained in Section III of the Utility Routines manual.) The necessary Easycoder statements are shown below.

EASYCODER
CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	MAR	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62 / 63 ... 80
				PROG	SUPERx
	C			@MSS	admode,cua,tua,
	L			1,2	cp,
				END	START

The parameters of the macro call statement are defined as follows.

Parameter	Description
x	The character assigned by the user to create a name for the Supervisor. If the Supervisor being created is the prime Supervisor, x must equal the character assigned at the time the bootstrap is generated. If no assignment was made at bootstrap generation time, x must equal 1. This is necessary because the mass storage bootstrap loads the Supervisor whose name corresponds to the character assigned at bootstrap generation time. Segment names for the Supervisor are generated automatically. No program in the residence file other than another version of the Supervisor may have a name whose first five characters are SUPER.
admode	The highest address mode of the machine on which the Supervisor is to run. Acceptable values are 3 or 4. If machine size is greater than 32K, this parameter should be 4 (blank = 3).
cua	Peripheral control unit address of the job control file device, which must be a card reader. The peripheral control unit address of the card reader is expressed as two octal characters (including all six bits). (Blank = 41.)

Parameter	Description
tua	Peripheral control unit address of the operator control file device (either a control panel or a console typewriter). The peripheral control unit address of a console typewriter is expressed as two octal characters (including all six bits). (Blank = control panel used.) A console typewriter is required for foreground/background information.
cp*	The central processor model of the machine on which the Supervisor is to run. (Blank = 121, 201-2, 1201, 2201, or 4201; 0 = 201; and 1 = 201-1.)

*This is parameter 12.

SECTION VII

FOREGROUND/BACKGROUND ENVIRONMENT

INTRODUCTION

The Supervisor can control the simultaneous execution of two programs in a manner generally called foreground/background operation. The Supervisor controls the sharing of central processor cycles between the two programs by allowing one to continue operation while the other is waiting for the completion of an input/output operation. This multiprogramming capability is an integral part of the Supervisor and is activated through a request by the operator at the time the Supervisor is bootstrapped.

The operation of the two jobs is completely separate and independent; when one job is completed, another job may be started even though another program may still be running.

The following paragraphs specify the details of foreground/background operation.

PRINCIPLE OF FOREGROUND/BACKGROUND OPERATION

The function of the multiprogram control portion of the Supervisor is to monitor the sharing of the central processor and main memory by two programs: one foreground program and one background program.

A program is often forced to wait for an input/output operation to be completed before it can proceed with further computation. This is because peripheral operations generally take a long time relative to central processor and main memory operations. The design of the Series 200 allows the central processor to continue during peripheral operations. Thus, a second program can be in main memory concurrently with the first program and can use the central processor for its own operations while the first program is waiting.

The concept of priority is also involved. An application may require fast response to some external event, but it may not require the central processor between events. If two programs are in main memory at the same time, the second program may use the central processor whenever the first priority program does not need it. However, some mechanism is necessary to ensure the priority program's gaining control of the central processor when required. This mechanism is called an "interrupt."

The Mod 1 (MSR) Supervisor can control the simultaneous execution of a foreground program and a background program. Only one foreground program and one background program may operate at the same time. The foreground program operates until it executes an input/out-

put instruction for a peripheral device which has the peripheral interrupt feature. It then yields the central processor to the background program by executing a call to the Supervisor, which turns on the background program. The background program operates until the input/output operation of the foreground program is completed. At this time the peripheral device signals the central processor to interrupt the currently running program (the background program). This interrupt causes the background program to be suspended and control to be returned to the foreground program. The foreground program continues processing until it issues (1) a PDT to a peripheral device which has its peripheral interrupt feature activated, and (2) a call to the Supervisor. Control then reverts to the background program pending completion of the foreground peripheral order, i.e., until an interrupt occurs. Interaction of the foreground, background, and supervisor programs is continuous, based upon the input/output order, the call, and the interrupt itself. Operation continues in this manner until either the foreground or background program terminates, at which point another foreground or background program (respectively) may be loaded and started.

The Supervisor with its multiprogram control capability and the Series 200 peripheral interrupt feature together control the sharing of the central processor and main memory cycles between the background program and the foreground program.

Definition of Foreground and Background Programs

Foreground and background programs must follow certain rules, as given on pages 7-7 and 7-9, and a precise definition of a foreground program or of a background program can only be made by giving those rules. However, the following statements are a general description of the characteristics of foreground and background programs.

A background program is a program that does not use the peripheral interrupt feature of the Series 200 and that should perform extensive computation so that it will not often need to wait for input/output operations.

A foreground program is a program that uses the peripheral interrupt feature for at least one device and that should perform little computation but much input/output operation.

JOB CONTROL IN FOREGROUND/BACKGROUND ENVIRONMENT

At any one time up to two programs may be residing in main memory and sharing the central processor. No more than one background and one foreground program can operate at one time.

The Mod 1 (MSR) system also has an automatic job sequencing capability; when one job is completed, the Supervisor automatically reads the job control file to determine what the next job is and to load and start it. In a foreground/background environment, the operator has the ability to control the sequencing of foreground and background jobs separately and independently. When

a foreground job is completed, another foreground job can be started independently of any background job that may still be running. Similarly, when a background job is completed, another background job may be started independently of any foreground job that may still be running. The operator may also choose to run a foreground program without a background program or vice versa.

EQUIPMENT COMBINATIONS FOR FOREGROUND/BACKGROUND ENVIRONMENT

The equipment requirements for foreground/background capability are listed in Section I.

The following table lists the combination of central processor model, optional instruction features, and console typewriters that meet the requirements for foreground/background operation.

Table 7-1. Equipment Combinations for Foreground/Background Environment

Central Processor	Features Needed	Console-Typewriters Usable	Location of Interrupt Button
111-3	1111	220-1	Control Panel
		220-3	Typewriter
121-4	1011	220-1	Control Panel
		220-3	Typewriter
121-0-4	1011	220-2 220-3	Typewriter
126-3	1011	220-1	Control Panel
		220-3	Typewriter
201	011 and 012	220-2 220-3	Typewriter
201-1	011	220-2 220-3	Typewriter
201-2	010	220-1	Control Panel
		220-3	Typewriter
1201 1251 2201		220-1	Control Panel
		220-3	Typewriter

WRITING A FOREGROUND PROGRAM

This paragraph describes two aspects of writing a foreground program and gives a simple example of a foreground program. First, a foreground program must operate the peripheral interrupt feature for certain peripheral devices (card reader, printer, communications control unit, etc.). Second, the foreground program must obey the rules imposed by the Mod 1 (MSR) system for communicating with the Supervisor.

Types of Foreground Programs

A data transcription program is one type of foreground program. Operations are performed on low-speed terminal devices, such as a card reader or a printer, when the programmer wants to allow use of the central processor during these relatively long peripheral operations. Data Conversion C, described in the Data Conversion A and C manual, Order number 231, is an example of this type of foreground program. If the programmer desires to write such a data transcription program himself, he must know how to program the peripheral interrupt feature and how to use it with his specific peripheral devices, as described in the Series 200 Programmers' Reference Manual and the various manuals on the specific peripheral devices.

A communications processing program is a second type of foreground program. Interrupts occur as a result of information coming in over communications lines, and the system must accept and process the data. Honeywell supplies two macro routine packages to assist the programmer in writing a communications processing program. These packages are described in Communications I/O C, Order number 439, and Message Mode Communications I/O C, Order number 667.

Foreground Program Communication with the Supervisor

All foreground programs must communicate with the Supervisor by using macro call statements to perform the functions listed below. They cannot perform the functions themselves, and they cannot communicate with the Supervisor in any other way.

RETURN MACRO CALL

A foreground program yields use of the central processor to the background program through the use of the Return macro statement. A foreground program should use the Return macro statement when it has no further use for the central processor and would otherwise be forced to wait for the completion of a peripheral operation or for a peripheral interrupt.

At the time of the next peripheral interrupt, the foreground program will be started at the location immediately following the Return macro statement.

The foreground program must be in interrupt mode when the Return macro routine is executed.

The Return macro call statement has the following format.

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	L			@PRIM	adi,adf,pre,	
2						
3						

adi - The address mode of the Supervisor; the same value as specified for parameter 01 when the Supervisor was specialized.
3 = address mode 3
4 = address mode 4

adf - The address mode of the foreground program at the point at which the macro statement appears.
3 = address mode 3
4 = address mode 4

pre - One to three characters which will be prefixed to all symbols in the macro statement to avoid duplication with other symbols in the program or with other occurrences of the macro routine.

SEGMENT LOAD MACRO CALL

A foreground program can load another segment through the Segment Load macro statement. The specified segment will be loaded and control will be returned to the location immediately after the macro routine.

The Segment Load macro routine may be executed in either interrupt or normal mode.

The Segment Load macro call statement has the following format.

EASYCODER
CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPER	MAR	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5 6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80	
1		L		@PSEG	adi, pre, seg, adf,	
2						
3						
4						

adi - Same as above.

pre - Same as above.

seg - 2-character name of the segment of the foreground program to be loaded.

adf - Same as above.

EXIT MACRO CALL

The Exit macro statement is used to return control to the Supervisor when the foreground program is completed. It is the equivalent for a foreground program of the program exit for a background program; that is, it is a normal end-of-job exit.

The foreground program must be in interrupt mode when the Exit macro routine is executed.

The Exit macro call statement has the following format.

EASYCODER
CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___.___ PAGE ___ OF ___

CARD NUMBER	TYPE	MAR	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1	L			@PEIM	adi,pre,adf,	
2						
3						
4						

adi - Same as above.

pre - Same as above.

adf - Same as above.

NORMAL MODE MACRO CALL

In certain applications the foreground program must be able to respond to peripheral interrupts very quickly. This is the case, for example, if data are coming in over a communication line very rapidly. In order to respond to the interrupt, the foreground program should perform some of its processing in the normal mode. The Normal Mode macro statement allows the foreground program to continue processing in the normal mode. The foreground program is continued in the normal mode at the location following the macro statement.

The foreground program must be in the interrupt mode when the Normal Mode macro routine is executed.

The Normal Mode macro call statement has the following format.

EASYCODER
CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___.___ PAGE ___ OF ___

CARD NUMBER	TYPE	MAR	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1	L			@PROPR	adi,adf,pre,int,	
2						
3						
4						

adi - Same as above.

adf - Same as above.

pre - Same as above.

int - Address at which the foreground program is to be started when the next interrupt occurs. This parameter must be a symbol. If it is omitted, the next interrupt will transfer control to the location following the last executed Return macro call.

Example of a Foreground Program

The flowchart given in Figure 7-1 illustrates a simple foreground program that reads punched cards and lists them on a printer (an example of a data transcription program). A background program is allowed to run while the card reading and printing operations are going on. The foreground program uses the peripheral interrupt only for the card reader. It is not necessary to use it for the printer, since the foreground program still allows the background program to operate concurrently. Usually the slowest device, in this case the card reader is assumed, is the device for which the peripheral interrupt should be used. The foreground program runs entirely in the interrupt mode, and it is started in the interrupt mode by the Supervisor when it is loaded.

First, the program turns on the Allow function for its card reader, using PCB instructions. It reads a card. Since this operation will take a relatively long time and since the foreground program cannot do any processing until the card has been read into memory, the foreground program issues a Return macro call to allow the background program to continue. When the card read operation is over, there is a peripheral interrupt, and the Supervisor returns control to the foreground program at the location following this Return macro call. The foreground program performs any necessary error tests. It then tests for end of file. If an end-of-file card has been read, the program turns off the Allow function for the card reader and performs an end-of-job exit using the Exit macro call.

If an end-of-file card has not been read, the card image is moved to the print buffer and is printed. The program now branches back to the coding to read the next card. Note that both the print operation and the card read operation are concurrent with background operations. Also, the card image must be moved out of the card input buffer before the next card is read.

PROGRAMMING CONSIDERATIONS FOR FOREGROUND/BACKGROUND ENVIRONMENT

Programs that operate in a foreground/background environment must observe the following rules. Note that a program may operate alone even if it does not follow all the rules for background programs; however, if it is to operate simultaneously with a foreground program, all the rules for background programs must be observed.

Rules for Both Foreground and Background Programs

1. The timing or speed of peripheral devices must not be assumed in the logic of a user's program.
2. The memory locations required by each program must be predetermined so that conflicts do not arise at execution time. If a background program uses additional memory at execution time, it must ensure that the additional memory allocated does not overlap the areas reserved for the Supervisor and the foreground program. The highest location available to background programs is stored in the Supervisor's HMA field, locations 187-189 (decimal).

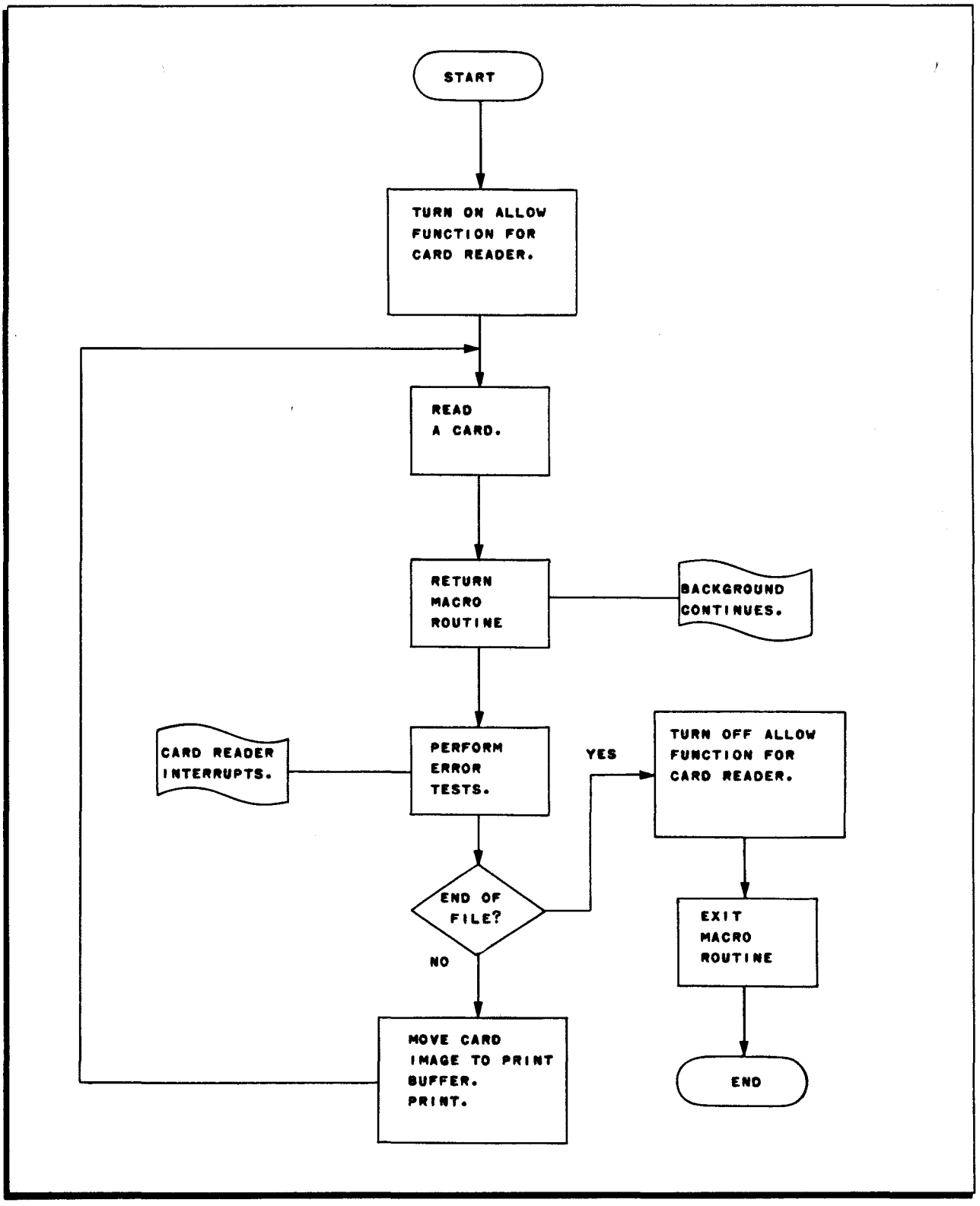

Figure 7-1. Example of a Foreground Program

3. The foreground and background programs cannot share read/write channels unless the background program does not rely on the settings of the read/write current and starting location counters that are shared.

4. Programs must not alter or refer to the interrupt register.

5. No background program can share a peripheral control unit equipped with the interrupt feature if any foreground program sets the interrupt allow function for that control unit.

6. Programs that have been converted using Easytran Symbolic Translator D cannot be run as foreground programs. However, programs of a background nature that have been converted by Easytran Symbolic Translator D can be run as background programs, provided that the foreground program does not use or disturb the change sequence register (CSR).

7. A background program must not use any read/write channels reserved to a foreground program.

8. The index register area (locations 1 through 60 decimal) must not contain record marks.

9. Mod 1 (Mass Storage Resident) system programs cannot operate concurrently with foreground programs under the Mod 1 (Tape Resident) system (that is, when using Interrupt Control D). In order to run with foreground programs, they must be run under the Mod 1 (MSR) Supervisor.

Rules for Foreground Programs

1. All foreground programs must set the interrupt allow function for their peripheral device(s). Normally, only terminal devices, that is, card reader, card punch, printer, or communications control units, should have the interrupt allow function set.

2. Foreground programs must turn off both the interrupt and allow functions for their device(s) before their normal end-of-job exit.

3. Foreground programs must do their own testing of device status; e.g., error testing.

4. Foreground programs must not communicate directly with the Supervisor; all communication must be made through the specified macro statements.

5. Foreground programs must not use index register X1 when processing data because this register is used exclusively for communication between a foreground program and the Supervisor. Foreground programs may use index registers X2 through X6.

6. A foreground program, if using the same mass storage peripheral control as the Supervisor or the background program, must save the mass storage address register at its common entrance point and restore it prior to issuing the Return macro call.

7. Foreground programs, using the same mass storage peripheral control as the Supervisor, and depending on the contents of the mass storage address register, should save the contents of the register prior to issuing a Segment Load macro call. The address register must be restored after the called segment is loaded. The locations in which the contents of the address register are saved should be different from those used for 6, above.

OPERATION IN FOREGROUND/BACKGROUND ENVIRONMENT

Operating procedures for a foreground/background environment differ slightly from those for a single-job environment. There are two areas of difference:

1. Bootstrapping the Supervisor - the operator decides whether or not foreground/background operation is desired.
2. Job Control - the operator controls the sequencing of jobs in a different manner.

The following paragraphs discuss these areas in detail.

Bootstrapping the Supervisor in Foreground/Background Environment

The operator has two additional considerations when bootstrapping the Supervisor. He must (1) choose between the single-job environment and the foreground/background environment and (2) decide upon the partitioning of main memory between the background and foreground programs.

CHOOSING THE FOREGROUND/BACKGROUND ENVIRONMENT

The operator must choose at the time the Supervisor is bootstrapped between the two possible operating environments:

1. Single job, where only one job may be executed at a time and there is no capability for operating a foreground (interrupt-driven or real time) program; or
2. Foreground/background, where both a foreground program and a background program may be run at the same time.

Once the operator has made this choice and the bootstrap process has been completed, operation of the Supervisor is automatic. The Supervisor need be bootstrapped again only if the operator wishes to change the environment or if the Supervisor has been destroyed in main memory.

CHANGING THE ENVIRONMENT

In the foreground/background environment, the Supervisor can control any one of the following combinations of jobs:

1. A foreground job together with a background job,
2. A foreground job alone, or
3. A background job alone.

If the Supervisor is operating in the foreground/background environment, the operator does not have to rebootstrap the Supervisor to operate in the single-job environment. The operator merely suspends processing of foreground jobs. However, the background program may be limited in the amount of main memory available to it, either by the setting of the limit

on background program memory or by the increased size of the Supervisor necessary for foreground/background operation.

In order to change from one environment to the other, the operator must perform the following steps.

1. Wait until all currently running jobs are completed (or terminate them manually, if necessary). In the foreground/background environment, both the foreground job and the background job must be completed or terminated.
2. Rebootstrap the Supervisor.

PARTITIONING MEMORY BETWEEN FOREGROUND AND BACKGROUND

When bootstrapping the Supervisor, the operator must consider the partitioning of memory between the background and foreground programs.

Certain programs can use to advantage additional main memory beyond their minimum requirement. For example, some system programs in the Mod1 (MSR) Operating System, such as Mass Storage Easycoder Assembler C, File Support C, and Mass Storage Sort C, can operate faster if more than the minimum 12,288 locations are available. These programs must be told the highest memory location they can use; otherwise, they might destroy the Supervisor, which resides in upper memory. The Supervisor's limit-of-object-program-memory field (also called the HMA field) specifies the memory limit. The HMA field is locations 187 through 189 in the Supervisor's communication area.

In the foreground/background environment, the foreground and background programs must share memory. Background programs that use additional memory must be told where the foreground program is so as not to destroy the foreground program. This is accomplished by partitioning memory between the foreground and background programs.

The background program is given the area from the Supervisor's communication area up to a "partition" location (specified by the user). The foreground program is given memory from this location up to the Supervisor. The Supervisor occupies the highest locations in main memory.

The operator must specify the location that is the partition between the background and the foreground programs. At the time the Supervisor is bootstrapped, the operator enters a character into memory that specifies the highest memory location that may be used by a background program. The Supervisor then places this indication appropriately into the HMA field. The Supervisor HMA field always reflects the highest memory location that may be used by a background program. Honeywell Mod1 (MSR) system programs observe this limit.

Note that a foreground program cannot use the HMA field to determine the location of the Supervisor.

Job Control in Foreground/Background Environment

Several aspects of job control are modified when the Supervisor is operating in the foreground/background environment. The main differences are as follows:

1. The ready state is reached manually by pressing the INTERRUPT button rather than by stopping the central processor. Stopping the central processor could interfere with a running real-time program.
2. The Supervisor provides additional information about the status of foreground/background operations. It informs the operator whether a foreground job has terminated.
3. Transition from one job to the next is not automatic. The operator is required to initiate the next job whether it is foreground or background.
4. The operator cannot alter the device through which the Execute statement is entered. A background program Execute statement is always entered through the card reader, a foreground program Execute statement is entered through the console typewriter.
5. The operator can perform additional manual operations. For example, he can terminate a background job at any time.

The Ready State

When operating in the foreground/background environment, the Supervisor is in the ready state whenever it is ready to accept a job control statement or action.

When the Supervisor is in the ready state, it types out the message READY on the console typewriter, requests a typein from the operator, and waits until that typein is made. The Supervisor does not continue automatically when a program has completed processing and exits.

The operator may perform only one action at the ready state. In order to perform a second action after the first action is completed, the operator obtains the ready state manually by pressing the INTERRUPT button.

The following paragraphs describe the methods of reaching the ready state when operating in the foreground/background environment.

BOOTSTRAP

The ready state is reached when the Supervisor is bootstrapped. The READY typeout appears.

The operator may initiate a job at this time by entering EX for a background program or EXF for a foreground program. He may also inform the Supervisor that there is no background job to initiate by entering TERM.

If he wishes to start a background program at a later time, he may do so by manually obtaining the ready state.

EXIT FROM A PROGRAM

The ready state is not reached when a program (either background or foreground) exits. The message END JOB B is typed to indicate the end of a background job; the message END JOB F is typed to indicate the end of a foreground job.

In order to initiate another job, the operator must obtain the ready state manually.

OBTAINING THE READY STATE MANUALLY

The operator can obtain the ready state manually by pressing the INTERRUPT button.

The operator can then perform any of the following actions.

1. Initiate a new job, either background or foreground.
2. Terminate a running background job and discontinue further background jobs.
3. Transfer control directly to the foreground program (containing Communications I/O C or Message-Mode Communications I/O C) by entering COMFG.
4. Enter spaces to tell the Supervisor to take no action, for example, if the operator has pressed the INTERRUPT button accidentally.

The Supervisor can control only one foreground program and only one background program simultaneously. If the operator requests initiation of a program and another program of that type is already running, the Supervisor will refuse to initiate the second program.

For each depression of the INTERRUPT button, only one action can be performed. If the operator desires to perform a second action, he must complete the first action before he presses the INTERRUPT button again.

Entering an Execute Statement

When operating in the foreground/background environment, there are two independent sources of job control information. To facilitate this independence, two separate job control

devices are used. Foreground programs must be requested through the console typewriter; background programs must be requested through the card reader. It is not possible to change these devices.

EXECUTE STATEMENT FOR A BACKGROUND PROGRAM

The Execute statement for a background program must appear in the card reader. The operator initiates a request for a background job by a console typewriter entry at the ready state; the Supervisor then reads cards until it encounters an Execute statement.

EXECUTE STATEMENT FOR A FOREGROUND PROGRAM

The Execute statement for a foreground program must be entered through the console typewriter. The operator initiates a request for a foreground job by a console typewriter entry at the ready state; the Supervisor then requests an entry for the name of the program segment to be loaded. This is done in the following manner.

After the typeout of READY, the operator must enter six characters representing the command field of a job control statement. If the operator wishes to initiate a foreground job, he must enter EXFΔΔΔ. Confirmation by the operator is performed in the standard manner by typing one space.

If the operator requested initiation of a foreground job, the Supervisor requests another typein immediately on the same line as the EXF. The operator continues by entering the operands field of an Execute statement. This entry is 49 characters (or less if a Type 220-3 Console Typewriter is used, since the operator can terminate the typein and verify it by pressing CARRIAGE RETURN). Verification is performed by typing spaces until the TYPE indicator goes off (or by pressing CARRIAGE RETURN with a 220-3).

The method of entering the Execute statement for a foreground job is identical to that used in the single-job environment with the console typewriter.

Communication Statement for a Foreground Program

The operator can initiate a request to transfer control immediately to the foreground program by means of the Communication statement for a foreground program, which must be entered through the console typewriter. Program control is transferred immediately to the instruction following the last executed Return or Normal Mode macro call. This is done in the following manner.

After the typeout of READY, the operator must enter six characters representing the command field of a job control statement. If the operator wishes to transfer control to the foreground program, he must enter COMFGΔ. Confirmation by the operator is performed in the standard manner by typing one space.

This statement can be used only when the foreground program currently running contains Communications I/O C or Message-Mode Communications I/O C.

RELATIONSHIP WITH PROGRAM TEST C

Program Test C cannot be run in the foreground/background environment.

SECTION VIII
OPERATING PROCEDURES

This section includes operating procedures for the Supervisor in the single-job and in the foreground/background environments.

LOADING A PROGRAM

Readying the Supervisor

In order for the Supervisor to be able to start a new job, it must be brought to the "ready state". In the ready state, the Supervisor is in a condition to process an Execute statement. In the single-job environment, a new job may be started. In the foreground/background environment, either a new foreground job or a new background job may be started unless a job of the same type is already running.

If the next Execute statement is to be read through the card reader in the single-job environment, the Supervisor automatically reads cards, searching for the Execute statement. If the Execute statement is to be read through the control panel, the Supervisor halts. If the Execute statement is to be read through the console typewriter, the Supervisor types out READY and waits for the operator to type in the Execute statement.

In the foreground/background environment, the operator always initiates the next job, whether foreground or background. (For further details, refer to the paragraph "Operator Control and Messages.")

The ready state can be obtained in one of the following ways.

BOOTSTRAPPING

The ready state is reached when the Supervisor is bootstrapped from mass storage. In the single-job environment, an Execute statement is read automatically (from the card reader). In the foreground/background environment, the operator must initiate jobs manually. The operator may initiate a job by entering EX for a background program or EXF for a foreground program. He may also indicate that there is no background program to initiate by entering TERM.

END OF JOB

In the single-job environment, the ready state is reached when a program reaches end of job (either normal or emergency). Normally, an Execute statement is read automatically;

however, if the operator has changed the device to the operator's console, he must then enter the Execute statement manually.

In the foreground/background environment, the ready state is not reached when a program reaches end of job. The operator must press the INTERRUPT button in order to obtain the ready state. He may then perform the actions listed below under "Manually Obtaining the Ready State."

MANUALLY OBTAINING THE READY STATE

The operator can obtain the ready state manually as described on Page 8-8 under "Manual Termination of a Job."

In the single-job environment, the ready state is reached just as for a (normal) end of job. See the above description.

In the foreground/background environment, the operator can perform any of the following actions.

1. Initiate a new job, by entering EX for a background job or EXF for a foreground job.
2. Terminate a running background job and discontinue further background jobs by entering TERM. (If a background job is to be initiated later, the ready state must be obtained manually at that time.)
3. Communicate directly with the foreground program (Communications I/O C or Message-Mode Communications I/O C) by entering COMFG.
4. Enter spaces. This tells the Supervisor to take no new action and to continue current job. This procedure would be employed if the operator had pressed the INTERRUPT button accidentally.

Entering the Execute Statement

The operator may enter an Execute statement in one of the following ways, depending on whether the single-job or foreground/background environment is in effect and whether (in the single-job environment) the operator has chosen the card reader or the operator's console (control panel or console typewriter). The paragraph entitled "Changing the Job Control Device," page 8-6, describes when and how the device used to read the Execute statement may be changed.

The format of the Execute statement is described in Section II. With minor variations, this format is used with whatever device reads the Execute statement.

CARD READER

If the device used is the card reader, the Execute statement for the next job (and any other job control statements for the job) should have been placed in the card reader. The Supervisor reads the job control file (in the single-job environment, this is done without pausing) and searches for an Execute statement.

If there are no job control statements in the card reader, the Supervisor comes to a stall condition, testing for card reader busy. The operator may place the Execute statement and other job control statements in the card reader and start the reader. The Supervisor then begins reading cards automatically.

NOTE: The operator does not need to press the STOP button on the central processor. The Supervisor resumes processing as soon as the card reader is ready.

CONTROL PANEL

If the device is a control panel, the Supervisor comes to a halt in the ready state. The operator enters the Execute statement through the control panel, as shown in Table 8-1, and presses RUN.

Table 8-1. Characters Necessary to Enter Execute Statement through Control Panel

Characters	Contents	Meaning
1 - 6	EX$\Delta\Delta\Delta\Delta$	Command field for an Execute statement.
7 - 15	ppppppss,	Name of the program segment to be loaded.
16 - 60	HALT=ppppppss,	Name of the segment to be loaded after which the Supervisor halts. This parameter is optional; if it is omitted, there is no halt. The halt-name parameter may be entered starting anywhere from character 16 on.

The A-address register contents define the location at which the Execute statement is to be entered. This address is the first location of a 60-character area into which the command and operands fields are to be entered.

CONSOLE TYPEWRITER

If the job control device is a console typewriter, the Supervisor types READY to indicate the ready state and awaits a typein by the operator. The operator enters the Execute statement through the typewriter as follows.

1. The operator enters the Command field.

Characters	Contents	Meaning
1-6	EXΔΔΔΔ	Command field for an Execute statement.

2. The operator examines his entry.

   a. If the entry was incorrect, he types on nonspace character. The Supervisor will return to the ready state.

   b. If the entry was correct, he types one space. The Supervisor will ask for a second typein.

3. The operator enters the operands field.

Characters	Contents	Meaning
7-15	ppppppss,	Name of the program segment to be loaded.
16-60	HALT=ppppppss,	Name of the segment to be loaded after which the Supervisor halts. This parameter is optional; if it is omitted, there will be no halt. The halt name parameter may be entered starting anywhere from character 16 on. However, it must contain no imbedded spaces.

4. The operator examines his entry.

   a. If the entry was incorrect, he types nonspace characters until the TYPE indicator goes off. The Supervisor will return to the ready state.

   b. If the entry was correct, he types spaces until the TYPE indicator goes off. (If a Type 220-3 console is being used, the operator may press the CARRIAGE RETURN key instead.)

   The Supervisor will proceed to interpret the Execute statement.

BOOTSTRAPPING THE SUPERVISOR

At the beginning of operations, the Supervisor must be bootstrapped from mass storage. Once the Supervisor has been bootstrapped, operation of the system is continuous. The Supervisor need be bootstrapped again only if it has been destroyed in memory. To bootstrap the Supervisor, proceed as follows:

1. Mount a mass storage volume containing the bootstrap routine and a residence file containing the Supervisor (the Supervisor is an executable program in the residence file) on a mass storage device assigned to drive number 0.
2. Press STOP.
3. Press INITIALIZE.
4. Set CONTENTS to lpp ppp, where pp ppp is the peripheral address assignment of the mass storage control.
5. Press BOOTSTRAP and wait for STOP to illuminate.
6. In a single-job environment, go to step 8. In a foreground/background environment, enter into location 363 (octal) the character F (26 octal).

7. If background programs are to be given 16K (16,384) characters of memory, proceed to step 8. Otherwise, enter into location 364 (octal) the bank indicator for the limit on background program memory. If the bank indicator is bb (octal), the highest address usable by background programs is bb7777 (octal). The value specified must be less than the relocation bank indicator in order to leave memory for a foreground program.

8. If the Supervisor being bootstrapped is the prime Supervisor (the Supervisor specified at bootstrap generation time), go to step 9. Otherwise, enter the last character of the Supervisor's name into location 367 (octal).

9. If the Supervisor being bootstrapped is to be located in the same memory bank as specified at bootstrap generation time, go to step 10. Otherwise, enter the relocation bank indicator into location 370 (octal). The values of the relocation bank indicator are as shown in Table 8-2.

Table 8-2. Values of Relocation Bank Indicator

Memory Size	Last Address Used by Supervisor (octal)	Relocation Bank Indicator (octal)
12K	027777	02
16K	037777	03
20K	047777	04
24K	057777	05
28K	067777	06
32K	077777	07
40K	117777	11
49K	137777	13
57K	157777	15
65K	177777	17
81K	237777	23
98K	277777	27
114K	337777	33
131K	377777	37
163K	477777	47
196K	577777	57
229K	677777	67
262K	777777	77

10. If the peripheral address assignment of the mass storage control is the same as specified at bootstrap generation time, go to step 11. Otherwise, enter the peripheral address assignment of the mass storage control into location 371 (octal) in the form 0pp ppp.

11. Press RUN.

The stated sequence of operations results in the Supervisor's being set to read an Execute statement, that is, the bootstrap operation brings the Supervisor to the ready state.

If a console typewriter is present, the bootstrap procedure is identical to that outlined above except for step 4, which is:

4. Press the "B" key followed by the peripheral control unit address of the mass storage control unit and six zeros, that is, B pp 000000.

Table 8-3 summarizes the adjustments that may be made at the time the Supervisor is bootstrapped. Each of the parameters can be adjusted at bootstrap time. If no change is made at bootstrap time, the column "Bootstrap Time Default" specifies the default value used. The column "Bootstrap Generator Default" indicates the value used if no value is specified to the Bootstrap Generator.

Table 8-3. Summary of Bootstrap Parameters

Parameter	Location (octal)	Possible Values and Meanings	Bootstrap Time Default	Bootstrap Generator Default
Environment indicator	363	F = foreground/ background not F = single job	not F	Cannot be specified to bootstrap generator
Limit of background program memory	364	bb (octal). Limit on background program is bb7777 (octal). This value must be less than the relocation bank indicator.	03	
Supervisor name	367	Any digit or letter	Uses bootstrap generator value (next column)	01
Relocation bank indicator	370	Any value from the relocation bank indicator table.		02
Residence file control unit address	371	pp- peripheral control unit address of the mass storage control unit.		04

CHANGING THE JOB CONTROL DEVICE

Normally, the next Execute statement must be read from the same device which contained the last Execute statement. The following paragraphs describe how to change the device used to read the Execute statement.

Single-Job Environment

In the single-job environment, the operator may change the device used to read the Execute statement. Normally, the device used is the card reader. In this case, the Supervisor

automatically searches for the next Execute statement in the card reader. If there is no Execute statement in the card reader, the Supervisor stalls on a card reader busy test. As soon as job control cards are placed in the card reader, the Supervisor will start up again.

The operator's console, instead of the card reader, can be used as the source of the Execute statement. The device used for the operator's console may be either the control panel or a console typewriter, depending on the choice made when the Supervisor was created.

The following paragraphs describe how to change the device used to read the Execute statement.

CHANGING FROM CARD READER TO OPERATOR'S CONSOLE

To change the device from the card reader to the operator's console, perform the following steps:

1. Ensure that cards are not in the card reader before the current program completes processing.
2. Press STOP on the central processor when the Supervisor stalls on the card reader busy test. If the STOP indicator does not illuminate, there is a read instruction stored in the card reader. It is necessary to place one or more blank cards in the card reader. (Do not press RUNOUT.)
3. Enter record marked 01 into location 100 (octal).
4. Place job control cards (excluding the Execute statement) in the card reader.
5. Press RUN..
6. At the 17002 halt or READY typeout, enter the Execute statement through the control panel or console typewriter.

CHANGING FROM OPERATOR'S CONSOLE TO CARD READER

To change the job control device from the operator's console to the card reader, perform the following steps:

1. If a console typewriter is being used and the READY typeout appears, press STOP on the central processor and type in seven spaces to release the console. If a control panel is being used, the 17002 halt will occur and the central processor will be in a halt state.
2. Enter record-marked 00 into location 100 (octal).
3. Place job control cards (including the Execute statement) in the card reader.
4. Press RUN.

Foreground/Background Environment

In the foreground/background environment, the operator cannot change the device used to read the Execute statement. The Execute statement for a background program must always come from the card reader, and the Execute statement for a foreground program must always be entered through the console keyboard.

ACTIVATING THE PROGRAM TEST FUNCTION

The program test function of the Supervisor is activated through job control statements for Program Development. Refer to the Program Development Subsystem manual. As a result of this activation, the Supervisor is altered to control Program Test. For example, the contents of the normal exit and emergency exit locations are modified; the size of the Supervisor changes; and this change is reflected in the high-memory-location-available field.

Program Test cannot operate in the foreground/background environment.

MANUAL TERMINATION OF A JOB

A running job, whether foregound or background, may be terminated by the operator. This process is essentially a manual method of performing an emergency end-of-job exit. The manner in which manual termination is performed depends on whether the Supervisor is in the single-job environment or the foreground/background environment.

Single-Job Environment

The operator may manually perform the emergency end-of-job termination as follows:

1. Press STOP, if necessary.
2. Set the sequence register to 00 01 26 (octal).
3. Set the central processor to the 3-character addressing mode.
4. Ensure that the sector bits of the A-address register are zeros.
5. Press RUN.

If Program Test C is being used, the emergency end-of-job dumps will be taken before the ready state is reached.

Foreground/Background Environment

In the foreground/background environment, the operator may desire to terminate either the background program or the foreground program.

MANUAL TERMINATION OF A BACKGROUND PROGRAM

A background program may be terminated only in one way, as described below. The STOP button must not be used, since it might interfere with a foreground program.

The operator terminates a background program as follows:

1. Press INTERRUPT. This brings the Supervisor to the READY typeout.
2. Type in TERM. Confirm the entry by typing spaces until the TYPE light goes off.

Any foreground program running concurrently will continue normally.

MANUAL TERMINATION OF A FOREGROUND PROGRAM

A foreground program may be terminated only by pressing STOP. The INTERRUPT button cannot be used. Any background program running concurrently cannot be continued.

OPERATOR CONTROL AND MESSAGES

At the time of a condition requiring operator action, information is conveyed to the operator as follows:

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

The operator takes corrective action and performs one of the following:

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 typewriter is being used.

Control Panel Operation

A control panel may be used only in the single-job environment; if the Supervisor is operating in the foreground/background environment, a console typewriter must be used for operator control.

When the operator control device is a control panel, information is conveyed to the operator through a Halt instruction. The B-address register indicates the meaning of the halt through a coded value. The A-address register may contain additional information.

NORMAL OPERATION HALTS

Table 8-4 lists the conditions that may be encountered during normal operation of the Supervisor.

Table 8-4. Normal Operation Halts

B-Address Register Contents	Meaning	Operator Action
000000	Bootstrap parameters may be entered. This halt is reached immediately after pressing the BOOTSTRAP button.	If any of the parameters has a nonstandard value, enter it into memory as described on page 8-5 and in Table 8-3.
017002	Supervisor requests an Execute Statement. This is the ready condition, i.e., with the Supervisor set to accept an Execute statement from the control panel.	Enter an Execute statement as described previously in this section.

Table 8-4 (cont). Normal Operation Halts

B-Address Register Contents	Meaning	Operator Action
014000	Halt-name program segment has been loaded.	Perform actions specified by operating instructions for the loaded segment. Press RUN.

ERROR CONDITION HALTS

Table 8-5 lists the conditions that may be encountered as the result of an error. Two of the items in Table 8-5 concern read errors. In these two cases if the associated device is mass storage, the A-address register displays six octal characters (18 bits) as follows:

A-Address Register

XXXXXX XXXXXX XXXXXX
123456

The setting of appropriate bits in the rightmost character of the A-address register indicates the following types of errors (bits are numbered from left to right, as shown above):

Bit 1 = Device inoperable

Bit 2 = Device error

Bit 3 = Immaterial

Bit 4 = Read error

Bit 5 = Instruction incomplete, and

Bit 6 = Immaterial

If no bits are set to one in the A-address register, the read error has occurred while the first bootstrap record is attempting to read the rest of the bootstrap routine.

Table 8-5. Error Condition Halts

B-Address Register Contents	Meaning	Operation Action
00pp10	Uncorrectable read error during bootstrap operation. pp = control unit address.	If the A-address register value is zero, repeat the bootstrap procedure. If other than zero, press RUN to try again.
017070	Supervisor requests relocation bank indicator.	Enter relocation bank indicator value into the location addressed by the A-address register.
014003	Bootstrap cannot find the residence file in the volume directory.	Repeat the bootstrap procedure with a volume containing the residence file.

Table 8-5 (cont). Error Condition Halts

B-Address Register Contents	Meaning	Operator Action
014002	Bootstrap cannot find the Supervisor in the residence file.	Repeat bootstrap procedure with a volume containing the proper version of Supervisor, or enter the proper Supervisor name when bootstrapping.
01pp10	Read error. pp = control unit address.	If the error is on the card reader, correct the last card read and press RUN to try again. If the error is on mass storage, press RUN to try again.
014004	First block of the segment on mass storage is not a beginning of segment block. Either the program being run has not computed an actual mass storage address properly and is using a "load by direct address"; or the residence file has been accidentally altered.	The program must be corrected or the residence file recreated. Repeat bootstrap procedure to go to next job.
014010	Segment called for was not found in the executable program file.	Repeat bootstrap procedure with correct volume containing segment called for.
017060	A foreground/background operation has been requested at bootstrap time, and the requested Supervisor does not have this capability.	Repeat bootstrap procedure changing Supervisor name, or press RUN to operate in single-job environment.

Console Typewriter Operation

A console typewriter must be used when the Supervisor is operating in the foreground/background environment. It may be used at the user's option when the Supervisor is operating in the single-job environment.

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

1. Read the typeout. If necessary, consult the manual for possible action.
2. Perform the desired corrective action.
3. Type the appropriate 1- character response (G, E, etc.).
4. If the typein is correct, press the space bar to continue. If incorrect, type any other character and return to step 3.

This procedure does not apply to the entry of an Execute statement.

Even though a console typewriter is used for communication with the operator, certain conditions encountered during the process of bootstrapping the Supervisor still result in halts. (This does not interfere with foreground/background operation, since the bootstrap process must be completed before concurrent running of foreground and background programs can start.) For this reason, three tables follow. Table 8-6 lists halts that result from conditions encountered during bootstrapping. Table 8-7 lists typed messages that result from normal conditions. Table 8-8 lists typed messages that result from error conditions.

BOOTSTRAP OPERATION HALTS

Table 8-6 lists the halts that result from conditions encountered during the process of bootstrapping.

Table 8-6. Bootstrap Halts

B-Address Register	Meaning	Operator Action
000000	Bootstrap parameters may be entered. This halt is reached immediately after pressing the BOOTSTRAP button.	If any of the parameters has a nonstandard value, enter it into memory as described on page 8-5 and in Table 8-3.
017070	Supervisor requests relocation bank indicator.	Enter relocation bank indicator value into the location addressed by the A-address register.
00pp10	Uncorrectable read error during bootstrap operation. pp= control unit address.	If the A-address register value is zero, repeat the bootstrap procedure. If other than zero, press RUN to try again.
014003	Bootstrap cannot find the residence file in the volume directory.	Repeat the bootstrap procedure with a volume containing the residence file.
014002	Bootstrap cannot find the Supervisor in the residence file.	Repeat bootstrap procedure with a volume containing the proper version of Supervisor, or enter the proper Supervisor name when bootstrapping.
017060	A foreground/background operation has been requested at bootstrap time, and the requested Supervisor does not have this capability.	Repeat bootstrap procedure changing Supervisor name, or press RUN to operate in single-job environment.

NORMAL OPERATION MESSAGES

Table 8-7 lists the typed messages that result from normal conditions.

Table 8-7. Normal Operation Messages

Message	Meaning	Possible Response	Meaning of Response
READY	Ready state.	EX	Initiate a background job.
		EXF	Initiate a foreground job.[1]
		TERM	Terminate background.
		COMFG	Transfer control to foreground program.[2]
		blanks	Perform no action.
ppppppss LOADED	Halt-name program segment has been loaded. ppppppss = program segment name of the unit just loaded.	G	Start the program segment.
		H	Halt (B-address = 14000)
		E	Emergency exit.
END JOB B	End of a background job.	None	-
END JOB F	End of a foreground job.	None	-
REFUSED	Incorrect or conflicting entry.	None	-

NOTES:

1. If the operator makes the entry EXF to request the initiation of a foreground program and if there is not a foreground program running already, a request will be made for a second typein (TYPE light goes on). The operator now enters the operands field of an Execute statement on the same line. See "Entering an Execute Statement" on page 7-13.
2. May be used only when Communications I/O C or Message-Mode Communications I/O C is being run in the foreground.

ERROR CONDITION MESSAGES

Table 8-8 lists the typed messages that result from error conditions.

Table 8-8 Error Condition Messages

Message	Meaning	Possible Response	Meaning of Response
pp d READ ERROR	Uncorrectable read error. pp = control unit address d = drive number	G	Reread.
		E	Emergency exit.
BEGINNING OF SEGMENT NOT FOUND	First block of the segment on mass storage is not a beginning of segment block.	E	Emergency exit.
ppppppss NOT FOUND	Segment called for was not found in the executable program file. ppppppss = program segment name	E	Emergency exit.

APPENDIX A

COMMUNICATION AREA

This appendix defines the Supervisor communication area. The following information is needed only if the programmer prefers to write his own instructions for communicating with the Supervisor, instead of using the calls that are provided for loading the next segment or exiting to the Supervisor.

The communication area for the Supervisor occupies location 0 and locations 61-189 (decimal). Through the communication area, the programmer supplies the necessary parameters to the Supervisor so that the functions of searching, loading, and starting can be performed. Table A-1 describes the communication area according to memory location. Table A-2 lists the fields of the Supervisor according to function. The expression "Initial Values" in Tables A-1 and A-2 refers to the contents of the corresponding field at the time the Supervisor is first brought into memory. "Reset" refers to a value entered into the field by the Supervisor at the time indicated, either just after a program exits or just before control is transferred to a loaded segment.

Punctuation marks with the communication area must not be altered. Each field is originally loaded with a word mark at its high-order or leftmost end.

Table A-1. Communication Area of the Supervisor

Locations		Permissible Values and Functions	Initial Value	Reset at Program Exit	Reset after Loading
Decimal	Octal				
61	75	Reserved for Supervisor use.			
62	76	Relocation bank indicator.	Determined by bootstrap generator		
63	77	Address of peripheral control from which Supervisor was bootstrapped.	Determined by bootstrap generator		
64	100	Device for Execute statement: 00 - Card reader 01 - Operator's console	00		
65-67	101-103	Revision number of unit last loaded.	Δ		

Table A-1 (cont). Communication Area of the Supervisor

Locations		Permissible Values and Functions	Initial Value	Reset at Program Exit	Reset after Loading
Decimal	Octal				
68-73	104-111	Program name (or mass storage address).	Δ		
74-75	112-113	Segment name.	Δ		
76	114	Reserved for Supervisor use.			
77-84	115-124	Halt name.	Δ		
85	125	Reserved for Supervisor use. (unique Supervisor identification).	Determined by bootstrap generator		
86-89	126-131	EMERGENCY EXIT.	Branch to Supervisor		
90-101	132-145	Reserved for Supervisor use.			
102-105	146-151	Exit to own-code routine.	Branch to Supervisor	Branch to Supervisor	Branch to Supervisor
106	152	Reserved for Supervisor use.			
107-109	153-155	Relocation augment.	000	000	000
110	156	Reserved for Supervisor use.	Δ		
111	157	Search mode: 20 - Search and load by program and segment name. 60 - Search and load by program and segment name and by visibility. 22 - Search by program and segment name; do not load. Supply mass storage address to calling unit. 62 - Search by program and segment name and by visibility; do not load. Supply mass storage address to calling unit. 07 - Do not search; load by known mass storage address. 01 - Decimally add a value of 1 to the segment name specified in locations 74 and 75; then proceed as described for search mode 20 (above).	20	20	

Table A-1 (cont). Communication Area of the Supervisor

Locations		Permissible Values and Functions	Initial Value	Reset at Program Exit	Reset after Loading
Decimal	Octal				
112	160	Start mode: N - Normal R - Return to calling program S - Special	N	N	
113-118	161-166	Visibility mask.	-00000 (visibility A)		
119-121	167-171	Special starting location.	000		
122-125	172-175	Own-code return before distribution.	Branch to Supervisor		
126-129	176-201	Own-code return after distribution.	Branch to Supervisor		
130-138	202-212	Segment load entrance, 3-character mode.	Store B-address register and branch to Supervisor		
139-141	213-215	Normal exit, 3-character mode.	Branch to ready state		
142-146	216-222	Current date.			
147	223	Trapping mode: 00-Off; 04-On.	0		
148-154	224-232	Reserved for Supervisor use.			
155	233	Operator control file device: 1. Control panel: item mark 2. Typewriter: word mark plus pcu address (6 bit).	Set when Supervisor is created.		
156-163	234-243	Reserved for Supervisor use.			
164-167	244-247	Normal exit, 4-character mode[1]			
168-171	250-253	Segment load entrance, 4-character mode[1]			
172-186	254-272	Reserved for Supervisor use.			
187-189	273-275	Limit of object program memory.			

[1] Available only if Supervisor has been specialized in 4-character address mode.

Table A-2 presents the fields of the communication area according to the functions with which each is used.

Table A-2. Summary of Supervisor Fields by Function

Function	Fields	Locations		Utilization
		Decimal	Octal	
Searching	Search mode	111	157	20 - Program and segment name. 60 - Program and segment name and visibility. 22 - Program and segment name; do not load. 62 - Program and segment name and visibility; do not load. 07 - Load by known mass storage address. 01 - Decimally add a value of 1 to the segment name specified in locations 74 and 75; then proceed as described for search mode 20.
	Program name	68-73	104-111	
	Segment name	74-75	112-113	
	Visibility mask	113-118	161-166	
Loading	Relocation augment	107-109	153-155	
	Halt name	77-84	115-124	
	Exit to own-coding	102-105	146-151	
	Own code return before distribution	122-125	172-175	
	Own code return after distribution	126-129	176-201	
Starting	Start mode	112	160	N = Normal S = Special R = Return
	Special start location	119-121	167-171	
	Trapping mode	147	223	00 = Off 04 = On

Table A-2 (cont). Summary of Supervisor Fields by Function

Function	Fields	Locations		Utilization
		Decimal	Octal	
Loading a Segment	Segment load, 3-character mode	130-138	202-212	B 130
	Segment load, 4-character mode	168-171	250-253	B(168)
Exiting	Normal exit, 3-character mode	139-141	213-215	B(139)
	Normal exit, 4-character mode	164-167	244-247	B(164)
	Emergency exit	86-89	126-131	B 86 or manually set the sequence counter to 86.
Supervisor Information	Relocation bank indicator	62	76	
	Address of control from which Supervisor was bootstrapped	63	77	
	Job control device	64	100	00 - Card reader 01 - Operator's console
	Supervisor identification character	85	125	
	Operator control file device: 1. Control panel: item mark 2. Typewriter: word mark plus pcu address (6 bit).	155	233	Set when Supervisor is created.
Other Features	Revision number	65-67	101-103	Revision number of segment last loaded.
	Current date	142-146	216-222	
	Highest memory location available	187-189	273-275	

APPENDIX B

COBOL C FLOATING CARD LOADER-MONITOR

FUNCTIONAL DESCRIPTION

COBOL C Floating Card Loader-Monitor, which is a self-loading deck of punched cards, loads and executes unsegmented punched-card programs produced by COBOL Compiler C in binary load deck (BLD) format. It loads these programs from the card reader in normal start mode, i.e., in the order in which they are to be executed. Depending on the console entry by the operator, COBOL C Floating Card Loader-Monitor can relocate itself in up to 32K characters of main memory.

EQUIPMENT REQUIREMENTS

The following equipment is needed.

1. A Series 200 central processor with advanced programming instructions.
2. Approximately 510 main memory locations, including decimal locations 025 through 189 (031 through 275 octal), and the last 345 locations of the bank to which the loader-monitor is relocated.
3. Index registers X5 and X6.
4. One card reader.

BOOTSTRAP AND LOADER OPERATING PROCEDURES

To bootstrap and load from cards, proceed as follows:

1. Place COBOL C Floating Card Loader-Monitor directly in front of the binary load deck (BLD). This combination of decks is called the binary run deck.
2. Insert the binary run deck in the card reader.
3. Press INITIALIZE.
4. Set the CONTENTS buttons to the octal address assignment of the card reader (normally 41 octal).
5. Set the ADDRESS buttons to 0000.
6. Press the BOOTSTRAP button. This causes the bootstrap card (first card of the binary run deck) to be read into memory starting at location 0.
7. Press RUN. This causes the instructions on the bootstrap card to be executed, resulting in the subsequent loading of COBOL C Loader-Monitor into memory.
8. When halt 17070 occurs, enter the relocation indicator.
9. The binary load deck following the loader is now read and loaded into memory. If SENSE switch 1 is ON, a programmed halt occurs after the deck is loaded. Press RUN to continue.

PROGRAMMED HALTS

Table B-1 shows the programmed halts contained in COBOL C Floating Card Loader-Monitor.

Table B-1. Programmed Halts for COBOL C Loader-Monitor

A Address	B Address	Meaning	Operator Action
xxxxxx	0xpp11	Read error.	Refeed card in error and press RUN to reread.
xxxxxx	014000	This halt occurs after the program has been loaded if SENSE switch 1 is ON.	Perform any actions requested by the programmer and press RUN to begin execution.
xxxxxx	017002	Current program has reached end of job, and control has returned to the loader.	Set up, initialize, and bootstrap loader-monitor so that the next program can be executed.
xxxxxx	017070	Halt for memory size entry.	If more than 12K characters of main memory are available and the programmer desires to use them, the operator must key one of the following entries into location 0. Keyin — Total Memory Available 3 — 16K 4 — 20K 5 — 24K 6 — 28K 7 — 32K Regardless of whether an entry has been made, press RUN to continue.

NOTES: pp = octal address assignment of loading device.
x = contents unspecified.

Honeywell

HONEYWELL

TECHNICAL PUBLICATIONS REMARKS FORM *

TITLE: MOD 1 (MSR) SUPERVISOR

DATED: OCTOBER, 1967

FILE NO: 123.5005.141C.1-616

ERRORS NOTED IN PUBLICATION:

Fold

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

Fold

(Please Print)

FROM: NAME _____ DATE _____

COMPANY _____

TITLE _____

ADDRESS _____

_____

Cut Along Line

* 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