SDS

SCIENTIFIC DATA SYSTEMS

Reference Manual

SDS Real-Time MONITOR

Price: $2.00

SDS REAL-TIME MONITOR REFERENCE MANUAL

900 SERIES/9300 COMPUTERS

February 1966

SDS

SCIENTIFIC DATA SYSTEMS/1649 Seventeenth Street/Santa Monica, California/UP 1-0960

Printed in U.S.A.

RELATED PUBLICATIONS

Name of Document	Publication No.
SDS SYMBOL and META-SYMBOL Reference Manual	90 05 06
SDS Business Language Reference Manual	90 10 22
SDS FORTRAN IV Reference Manual	90 08 49
SDS FORTRAN IV Operations Manual	90 08 82
SDS 910 Computer Reference Manual	90 00 08
SDS 920 Computer Reference Manual	90 00 09
SDS 925 Computer Reference Manual	90 00 99
SDS 930 Computer Reference Manual	90 00 64
SDS 9300 Computer Reference Manual	90 00 50

CONTENTS

1. INTRODUCTION 1

Features 1
General Description 1
Hardware Requirements 2

2. MONITOR CONTROL MESSAGES 3

System Control Messages 3
Processor Control Messages 5
Sequential File Subcontrol Messages 8
Input Control Messages 8
Operator Control Messages 8

3. LOADING 10

Batch Jobs without Overlay 10
Batch Jobs with Overlay 10
Loading to and from Secondary Library 11
Loading Resident, Real-Time Programs 11

4. PREPARING THE PROGRAM DECK 12

META-SYMBOL Initial Assembly 12
META-SYMBOL Assembly with Corrections 12
FORTRAN Compilation 12
FORTRAN Compile-and-Execute 12
Overlay Program Example 13
Multiple Program Example 14
SYMBOL Program Example 15

5. REAL-TIME OPERATIONS 15

6. DEBUGGING 16

7. OPERATOR ACTIONS 17

8. MONITOR INTERFACES 18

I/O Operations 18
I/O Programming 19
Magnetic Tape Operations 21
Card Reader Punch Operations 21
Line Printer Operations 22
Paper Tape and Typewriter Operations 22
Disc File Operations 23
MONITOR Subroutines 24
Standard Calling/Receiving Sequences 27

9. SYSTEM UPDATE ROUTINE 32

General Description 32
Update Processor Control Messages 32

10. SYSTEM GENERATION 33

General Description 33
SYSGEN Control Messages 33
System Standard Assignment Generation 34
Standard Assignment Modifications 34
User Hold Files (Disc) 34
General System Generation 35
Summary of SYSGEN Messages 35

INDEX 48

APPENDIXES

A SDS STANDARD BINARY LANGUAGE 41

B 900 SERIES REAL-TIME TAPE MONITOR 47

1. INTRODUCTION

The Real-Time MONITOR[†] for SDS 900 Series and 9300 Computers is a comprehensive system for monitoring and controlling assemblies, compilations, and other program operations. Some of its outstanding features are:

- efficient system operation with minimum operator intervention;
- an easy-to-use, on-line, real-time input/output facility having maximum efficiency, while taking into account the needs of the user's program (I/O operations are performed simultaneously with the user's program);
- an open-ended set of processors that include the SDS META-SYMBOL assembler and SDS FORTRAN IV; and
- a system of diagnostic routines, including highly selective program dumps.

Rapid Access Disc (RAD) Files are used as the storage medium for MONITOR, the processors that it controls, library routines, and system scratch and user files. MONITOR is also available in a magnetic-tape version, for use with systems that do not have RAD Files. If the tape version of MONITOR is used, loading to the secondary library cannot be accomplished by MONITOR, since the secondary library must be created at system generation time. Also, any references (in this manual) to HOLD files, sequential disc files, and random access disc files are not applicable to the tape version of MONITOR. All other facilities, however, are provided (see Appendix B).

All major elements of MONITOR are stored on the RAD Files in such a way that they are always available to the system. Accordingly, these elements are said to be system resident. The system resident feature, among others, enables MONITOR to operate in real-time, providing the user with a time-sharing environment in which to solve both real-time and nonreal-time problems. The full real-time processing capability of the hardware is utilized through interrupt control, with a minimum of effort on the part of the user. Batch processing functions, such as the compilation of FORTRAN IV programs and the assembly of SYMBOL/META-SYMBOL programs, are provided also. Both forms of processing may be carried out intercurrently, i.e., allowing interrupt service routines to be resident while batch processing functions are carried out, preempting control when an interrupt occurs, and restoring control to the batch processing function when interrupt servicing is completed.

FEATURES

BATCH PROCESSING

Except for the servicing of interrupts, all operations are considered to be in the batch processing mode. A batch processing function may be "swapped out" if its storage is needed by an interrupt service routine. An interrupted function will resume execution from the point of interrupt, after the interrupt has been serviced completely. A portion of RAD storage is reserved for the "swap out" operation.

Files used during batch processing are protected from destruction by interrupt service routines for the duration of the job. Conversely, files that have been reserved explicitly for interrupt servicing may not be accessed by batch processing functions. When interrupt service routines are resident in core memory, batch processing jobs are prohibited from doing "compile and go" operations. This is done to protect resident real-time programs from destruction by batch job execution.

REAL-TIME PROCESSING

Any interrupt service routine may be made resident in core storage if sufficient storage is available. References to primary library routines within such resident programs are satisfied at the time that they are loaded. References to other routines are satisfied at execution time, by dynamically loading the referenced program into unused storage. Such a dynamic load may cause an interrupted batch job to be "swapped out" if sufficient storage is not otherwise available. Routines that have been dynamically loaded may themselves cause dynamic loading of other routines.

Core storage used by resident interrupt service routines can be protected through use of the optional Memory Protection Feature.

DISC FILES

The user may define his own disc files. Such files may be either random access or sequential. Sequential files can be considered to be simulations of magnetic tape files. Consequently, it is possible to have essentially device-independent files defined in a program, and to determine the device at the time of execution.

GENERAL DESCRIPTION

The main elements in the MONITOR system are a resident monitor, a FORTRAN IV processor, a SYMBOL assembler, a META-SYMBOL assembler, an overlay loader, a memory dump program, an I/O processor, a primary library, a secondary library, and an update program.

RESIDENT MONITOR

The resident monitor consists of an executive, an interrupt monitor, a reentrance monitor, a system bootstrap program, and a resident loader.

Executive

The executive is the central system control, and processes all control messages. Provision is made, in the executive, for symbolic access to resident user programs.

[†]Throughout this document, the entire system will be referred to as MONITOR, as distinct from "the monitor" which is a primary control portion of the system.

Interrupt Monitor

The interrupt monitor controls all interrupt processing. When an interrupt occurs, the interrupt monitor saves hardware and program status information and performs other functions before turning control over to the appropriate interrupt service routine. After completion of the interrupt service routine, the interrupt monitor restores the previous program and hardware status in the computer, clears the interrupt, and returns control to the point in the program at which the interrupt occurred. In addition to program status information, each interrupt save block includes the contents of the following registers:

1. A register
2. B register
3. X register(s)
4. FORTRAN double-precision register
5. FORTRAN complex register
6. Location 0 (900 Series only)

Interrupt save blocks are placed in dynamic storage, and are restored following the servicing of each interrupt.

The interrupt monitor is itself interruptable and reentrant. All pointers, counters, etc., that are interrupt-level dependent are contained withing the interrupt monitor as temporary storage locations (temps), and are thus saved whenever the interrupt monitor is interrupted. These include storage allocation pointers, the interrupt level counter, reentrance list pointers, and so forth.

Reentrance Monitor

The reentrance monitor determines whether or not a subprogram is being reentered as the result of an interrupt. When this is the case, the reentrance monitor saves the previous status of the routine before allowing it to be reentered. The local variables, temps, return address, calling argument addresses, etc., are stored in a push-down list. Each item in the reentrance push-down list is chained to the previous item, so that the interrupt monitor can scan the list when operations are complete at the end of each interrupt level and restore the information saved at that level.

System Bootstrap Program

The system bootstrap program is an absolute program that can read itself into core memory from disc storage. It causes the MONITOR to be read into core memory and causes the executive wait loop to be entered.

Resident Loader

The resident loader consists of a loader control program, a semiabsolute loader, an implicit call processor, and various utility routines. It is used to load all programs from the system files, including programs from the primary library, the secondary library, and the overlay file.

REAL-TIME FORTRAN IV

The FORTRAN IV processor will operate in real-time and is ASA compatible.

META-SYMBOL ASSEMBLER

The META-SYMBOL assembler translates source programs, written in symbolic code, into machine language programs.

SYMBOL ASSEMBLER

The SYMBOL assembler is similar to the META-SYMBOL assembler, but without some of the latter's advanced features.

OVERLAY LOADER

The overlay loader converts relocatable programs into a form suitable for loading. It allows the loading and operating of a program in segments.

MEMORY DUMP PROGRAM

The memory dump program can be called during batch processing as the result of a control message. A dump may be taken of any specified area of memory and the information dumped is in octal format.

SYSTEM I/O PROCESSOR

The system I/O processor is a general I/O package that is used to process all input/output functions, including system requests. All I/O operations occur on a first-come-first-served basis; and an operation, once started, continues to completion. Prior to the start of any I/O operation, an interrogation is made to determine whether or not the physical file has been reserved. If the file has been reserved, and if the request is from a resident program, the request is serviced; otherwise, an error message results.

PRIMARY LIBRARY

The primary library consists of user routines such as mathematical routines and FORTRAN I/O and system routines. Other routines may be added at the user's discretion. A debug program is provided as a standard feature and includes such facilities as tracing, patching, snapshots, memory display, and address search.

SECONDARY LIBRARY

The secondary library includes user routines such as interrupt service routines and batch processing production programs.

UPDATE PROGRAM

The update program, accessed by means of control messages, has the ability to perform the standard file maintenance functions of insertion, deletion, and replacement. It may be used for both system and user file updates.

HARDWARE REQUIREMENTS

MONITOR will function in any SDS 900 Series/9300 Computer having at least 16K words of core memory. The META-SYMBOL assembler requires 12K words of memory. If there are to be no real-time resident programs and if META-SYMBOL is not to be used, the minimum required memory for MONITOR is 8K words. If real-time programs are to be in residence, at least 12K words are required.

The system requires interlace on each channel or buffer controlling:

1 Typewriter
1 Magnetic Tape Unit
1 Card Reader
1 Rapid Access Disc File (524K characters.)

MONITOR has provisions for using the optional memory protection, multiple memory bank, and power-fail-safe features.

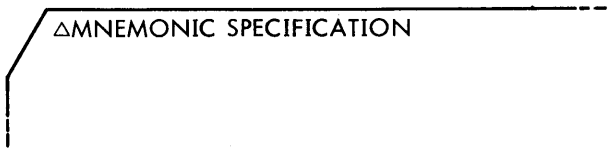
2. MONITOR CONTROL MESSAGES

The user directs and controls MONITOR via control messages. These messages direct the construction and execution of programs and provide the link between the program and its environment. The environment includes MONITOR and its processors, the computer operator, and peripheral equipment.

The control messages are

System Control	Loader Control
JOB	SEG
ASSIGN	INCLUDE
RELEASE	
DATE	Sequential File Subcontrol
TITLE	
MESSAGE	BACKSPACE
LABEL	REWIND
PAUSE	ENDFILE
Processor Control	**Operator Control**
METAXXXX	ABORT
SYMBOL	GO
FORTRAN	RETRY
LOAD	ERROR
	CONNECT
Input Control	
EOF	
DATA	
FIN	

Control messages have the general form

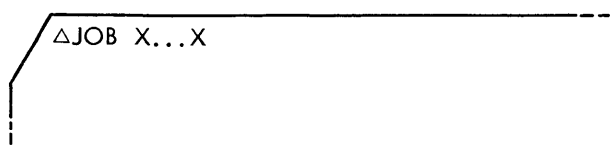


A space must follow the mnemonic, and no spaces are allowed within the mnemonic.

Columns 73-80 are not interrogated and may be used for identification.

SYSTEM CONTROL MESSAGES

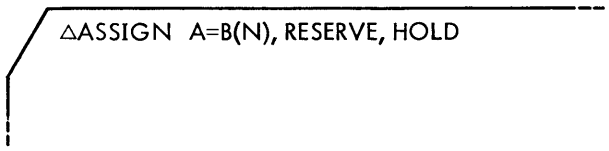
JOB JOB signals the completion of the previous job and the beginning of a new job.



X...X is any field desired by the user. Provision is made for the insertion of installation accounting routines, i.e., those monitoring the running characteristics of the jobs performed such as running time, units used, output quantity produced, etc.

When a job message is encountered, all user's operational labels (relating to a previous job), are deleted from the operational table.

ASSIGN ASSIGN provides for equating logical peripheral device names, designated by the user, to physical peripheral device names permanently established for the system. The operator normally generates ASSIGN cards from programmer-supplied job request information.



A is a symbolic file name of from 1 to 8 characters, designated by the user.

B is either a symbolic file name or else the permanent symbolic unit name of an attached physical unit. If B is a symbolic file name, it must have been defined previously.

(N) is a numeric constant which specifies a maximum size, in words, of disc storage to be allocated to this file. A size must not be specified for files not assigned to disc storage or for sequential disc files. A size must be specified for all random-access disc files. If B is a symbolic file name, (N) must not be specified.

RESERVE may be used when assigning files to be used by real-time programs only. The RESERVE option specifies that the file named may be read or written by real-time programs only.

HOLD applies to disc storage only. If a file is assigned to disc storage, it will become a permanent file whenever HOLD is specified and will remain on disc storage until released by a △RELEASE message.

B	Device
MD	Magnetic drum
MT	Magnetic tape
CP	Card punch
CR	Card reader
PP	Paper tape punch
PR	Paper tape reader
TY	Typewriter
LP	Line printer
DF	Disc file
PL	Plotter
NO	No I/O desired

The user may fill an ASSIGN card with as many complete assignments as will fit; no continuation is allowed.

Examples:

△ASSIGN ABBC=MT2A, QRZ=ABBC, 2=DF1A(5000)

Assign the magnetic tape 2 on channel A to the label ABBC, assign the label ABBC to the label QRZ, assign disc file 1 on channel A to the label 2 and allocate 5000 words of disc storage.

△ASSIGN ABC=MT3A, TYP=TY1A

Assign the magnetic tape 3 on channel A to the label ABC and the typewriter 1 on channel A to the label TYP.

Note that magnetic tape units are numbered 0 through 7; all other devices are numbered from 1.

MONITOR-defined system labels may be used as labels in ASSIGN control messages.

The complete set of standard file names that may be used with ASSIGN control messages (or RELEASE control messages explained below) are as follows:

System Labels

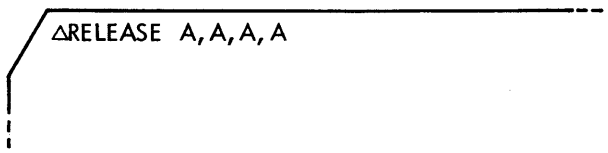
Label	Reference	Reassignable?
R\PROC	Processor file	No
R\PROK	PROC deck file	No
R\PRIL	Primary library file	No
R\SECL	Secondary library file	No
R\OVRL	Overlay library file	Yes
R\PERM	Permanent file	No
R\CONS	System typewriter	No
R\SWAP	Swap file	No
R\DUMP	Post-mortem-dump file	Yes
C	Control message input	Yes
Xi	Scratch files, where i = 1, 2, 3, . . .	Yes
SI	Symbolic input	Yes
SO	Symbolic output	Yes
EI	Encoded input	Yes
EO	Encoded output	Yes
LO	Listing output	Yes
GO	Binary output for load-and-GO	Yes
BI	Binary input	Yes
BO	Binary output	Yes
TY	Typewriter	Yes
NO	No I/O operation	No

Assignment, release, and reservation of files is accomplished through the use of File Control Blocks (FCBs) which communicate information about files to the system I/O handler.

Reservation of a file by MONITOR is accomplished by setting the reserve bit in the FCB for that file. Reservation of files not assigned to disc storage results in a reservation of the physical unit being assigned.

MONITOR accomplishes the reservation of files assigned to disc storage by allocating a disc storage area, of the required size, and adding the FCB to a table of FCBs on disc.

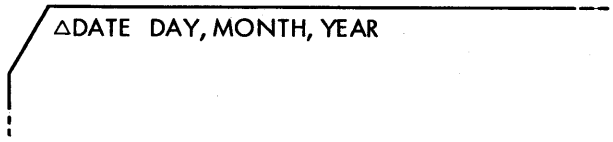
RELEASE RELEASE pertains to HOLD disc files or to those files that have been reserved.



A is a symbolic name of from 1 to 8 characters.

RELEASE instructs MONITOR to release the specified file(s) from its previous assignment.

DATE DATE gives MONITOR the date to be used for heading outputs. The date is also listed on the LO (listing output) media after each JOB card. (The assignment of a device as the one on which listing output is to be produced is explained under "Processor Control Messages.")



DAY is a 1- or 2-digit number.

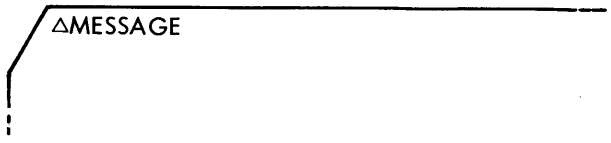
MONTH is a 3-letter abbreviation; if it is expressed, DAY must also be expressed.

YEAR is a 4-digit number; if it is expressed, MONTH must also be expressed.

TITLE The TITLE control message may appear anywhere after a JOB message and before a LOAD message. The purpose of the TITLE control message is to produce a heading at the beginning of each page on the LO media. MONITOR begins counting headed pages whenever a TITLE control message appears, and it incorporates this information into the heading.

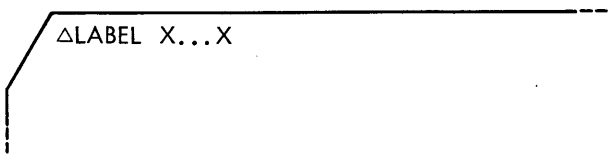
The contents of columns 9 through 72, the current date, and the number of the page appear at the beginning of each new page on the LO media.

MESSAGE The MESSAGE control message may appear anywhere.



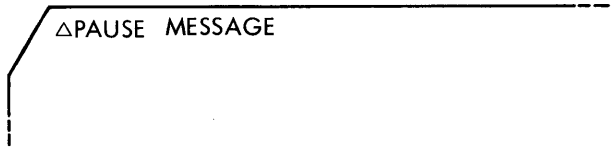
The contents of columns 1 through 80 are output on the system console typewriter and on the TY device (if not the same as the system console), and LO devices specified by processor control cards.

LABEL The LABEL control message may appear anywhere before the processor and LOAD control messages. The LABEL control message enables the user to write a label on the GO file (i.e., the file used to store the binary listing output in assemble-and-execute or compile-and execute processing) preceding the binary object code.



X...X is the label to be written on the GO file. It may be up to 8 characters in length.

PAUSE The PAUSE control message causes MONITOR to wait for an operator response before continuing, allowing time for the operator to perform some manual function (such as changing a tape reel).



This control message will cause the message

PAUSE TYPE △ABORT, △GO

to be produced on the system console preceded by the MESSAGE, and will cause the program to loop until the operator responds. (See Operator Control Messages.") It may occur anywhere.

PROCESSOR CONTROL MESSAGES

Processor control messages tell MONITOR what system, such as FORTRAN, is to be used with the input deck to follow. Any processor message also contains a list of input and output specifications to be used during the assembly or compilation.

The complete set of I/O specifications that are recognized by MONITOR are given in the following table.

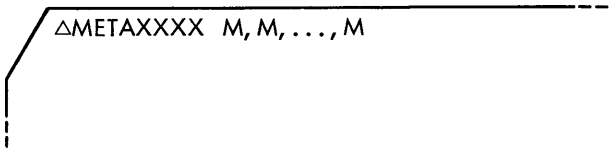
I/O Specifications

M	Specification Reference
EI	Encoded input
EO	Encoded output
SI	Symbolic input
SO	Symbolic output
LO	Listing output or listing object

I/O Specifications (cont.)

M	Specification Reference
LS	Listing source
BO	Binary output
X	Compile X cards
GO	Binary output for load-and-GO
S	SYMBOL-type symbolic statements
C	MONARCH compatibility
ASA	ASA standard storage allocation
CONC	Standard concordance listing
EXCP	Concordance listing with exceptions
910 920 925 930	Defines CPU for which code is to be generated. (900 Series FORTRAN IV only.) If unspecified, generated code will be for CPU on which compiler is operating.

METAXXXX METAXXXX specifies to META-SYMBOL which type of inputs and outputs the program requires.



XXXX is 920, 9300, or 910. META920 produces output for the 920/930. META9300 produces output for the 9300 and META910 produces output for the 910/925.

M is the input/output specification.

META-SYMBOL I/O Specifications

M	Specification Reference
EI	Encoded input
SI	Symbolic input
LO	Listing output
GO	Binary output for load-and-GO
BO	Binary output
EO	Encoded output
SO	Symbolic output
C	MONARCH compatibility
CONC	Concordance listing (standard)
EXCP	Concordance listing with exceptions

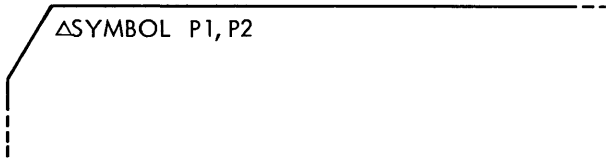
With the METAXXXX message:

The user writes the specifications (M, M, . . . , M) separated by commas, in any order.

Once established, a set of options remains in force throughout the job until changed by a new processor control message.

Note: If the encoded and symbolic input are from the same source, a second scratch file for META-SYMBOL is required.

SYMBOL The SYMBOL control message directs MONITOR to load and transfer control to the SYMBOL processor.

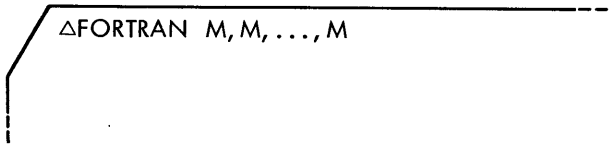


P1 is 920, 9300, or 910 (920 produces output for the 920/930, 9300 produces output for the 9300, and 910 produces output for the 910/925).

P2 specifies output data parameters. The parameters (LO and/or BO) may appear in any order, separated by a comma.

At least one output parameter (P2) must be present. Since symbolic input is assumed, SI is not used as a parameter.

FORTRAN The FORTRAN message informs MONITOR that the FORTRAN IV compiler is to be used to process the source deck.



M specifications, separated by commas, may be written in any order and have the configurations and meanings given in the table of FORTRAN IV I/O Specifications.

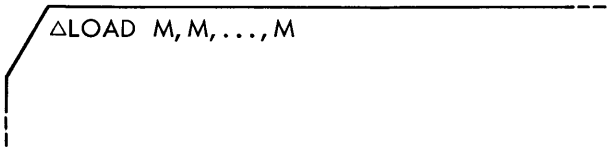
With FORTRAN, symbolic input (SI) is always assumed. If the user requests the LO option together with the LS option, the listing occurs in the order: source then object. The label X refers to FORTRAN IV conditional compilation (X in column 1) cards. The presence of the label X causes FORTRAN to compile X cards. Otherwise, it treats them as comment cards.

FORTRAN IV I/O Specifications

M	Specification Reference
SI	Symbolic input
BO	Binary output
LS	Listing source
LO	Listing object
ASA	ASA standard storage allocation
X	Compile X cards
GO	Binary output for load-and-go
S	SYMBOL-type symbolic statements occur on "S" cards in a FORTRAN program
SO	Symbolic output (for 9300 FORTRAN only)
910 920 925 930	Defines CPU for which code is to be generated. If unspecified, generated code will be for the CPU on which the compiler is operating. (900 Series FORTRAN only.)

Several FORTRAN programs can be compiled without preceding each one with a FORTRAN control message. Each subsequent FORTRAN program uses the same M specifications encountered in the last FORTRAN control message.

LOAD The LOAD control message causes MONITOR to use the loader to load programs.



Specifications for LOAD Control Messages

M	Specification Reference	Precludes Use of Specifications
'name'	Load named routine from secondary library into core	SECLIB
UPPER	Load from input medium into upper residence core area	LOWER, SECLIB, X, XM, XR, M100
LOWER	Load from input medium into lower residence core area	UPPER, SECLIB, X, XM, XR, M100
SECLIB	Load from input medium (except secondary library) into secondary library	'name', UPPER, LOWER, X, XM, XR, M100, BI
X	Load from input medium into core and execute only if errorless	UPPER, LOWER, SECLIB, XM, XR
XM	Load from input medium into core and execute only if no major errors	UPPER, LOWER, SECLIB, X, XR
XR	Load from input medium into core and execute regardless of errors	UPPER, LOWER, SECLIB, X, XM
M100	Origin of relocatable programs to be a multiple of octal 100 (except for overlay and library routines)	UPPER, LOWER, SECLIB
MAP	Produce load map	
BI	Search BI file for referenced 'name'; if not found, search SECLIB. 'name' must be specified if BI is used. If 'name' cannot be found on BI or SECLIB, an error message will be printed	SECLIB

M specifications, separated by commas, may be written in any order. The various M specifications are shown in the table on the preceding page.

The 'name' used as a specification in a LOAD control message must be the defined name of a routine in the secondary library. If a 'name' is specified, the specification SECLIB must not be used in the same LOAD control message. An example of the use of a 'name' is given below.

```
△LOAD X,'name'
```

The above message would cause the named routine to be loaded from the secondary library into core memory and executed only if free from errors. The lack of an X specification (either X, XM, or XR) would cause the named routine to be loaded into core memory but not executed.

If no 'name' is specified, MONITOR loads from the GO file if the GO file contains any binary code to be loaded. If the GO file contains no binary code, MONITOR then loads from the BI file. An example is given below.

```
△LOAD UPPER,MAP
```

The above message would cause the binary code from the input medium (i.e., the GO file, if possible, or else the BI file) to be loaded into the highest available resident core area. No execution of the loaded binary code would result, but a listing, or "map" of the relative locations of all external definitions would be printed out on the system console typewriter at load time. If the specification LOWER were used, rather than UPPER, the binary code would be loaded into the lowest available area of resident core memory instead of the highest. Note that the specifications UPPER and LOWER must not be used in conjunction with any of the X specifications, since resident core storage is not intended to be used for batch processing.

If SECLIB is specified, the binary code from the input medium is loaded into the secondary library. An example is given below.

```
△LOAD SECLIB,MAP
```

Note that the only other load specification that may be used in conjunction with SECLIB is MAP (when a load map printout is desired).

The specification BI may be used in conjunction with any other load specification except SECLIB. When BI is specified, a 'name' must be specified also. The named routine is loaded from the BI file or, if it is not found there, from the secondary library.

On completion of loading from the input source, the loader attempts to fulfill any unsatified references by searching the primary library. The routines that satisfy such references are added to the program. All other references are treated as implicit calls, i.e., references to undefined external symbols at load time (see "Implicit Call Processor").

LOADER CONTROL MESSAGES

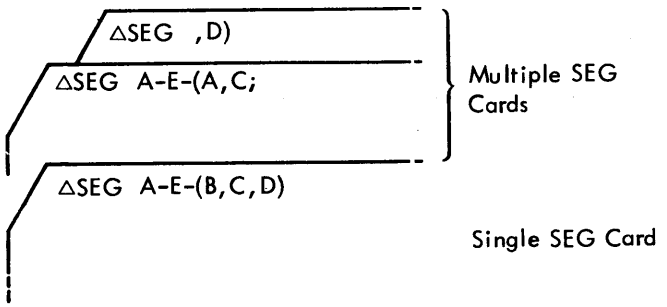
SEG A segment of a program is that portion of memory that is committed by a single reference. A segment usually overlays some other segment and is constructed from subsegments. A fixed segment is that portion that resides in core memory at all times. Any number of SEG cards may be used to define the program, but they must be in sequence. (See "MULTIPLE SEG CARDS".)

Segmentation may be specified by use of the following symbols on a SEG control card:

labels	one to eight alphanumeric characters that are the labels of segments.
-	indicates that two segments or segment levels are to be consecutive in memory.
,	indicates that two segments are to overlay each other (begin at the same point).
( )	indicates a grouping.
;	indicates that another SEG card follows.

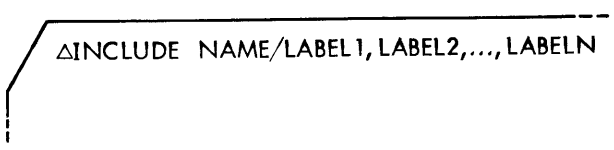
MULTIPLE SEG CARDS

The special terminator (;) is used to continue from one SEG card to another.



Comments may appear on a SEG card, provided that a semicolon is used to terminate the segmentation codes and the comments field begins at the right of the semicolon. Comments may not appear on the last SEG card.

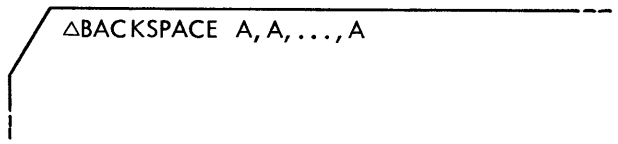
INCLUDE The loader normally allocates labeled COMMON blocks in the highest levels in which they are referenced. Library routines are usually loaded at the highest level (see Section 3). However, the user may allocate labeled COMMON blocks and library routines at any subsequent level by the use of the INCLUDE control card.



NAME	is the 1 to 8-character label used to define the segment in which the blocks or routines are to appear.
LABEL	is the label used to define the routine or COMMON block.

SEQUENTIAL FILE SUBCONTROL MESSAGES

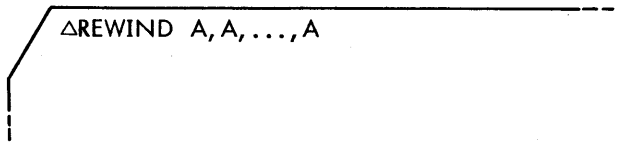
BACKSPACE BACKSPACE pertains to magnetic tape or sequential disc files only.



A is a symbolic name of from 1 to 8 characters representing a file on a magnetic tape or disc file (sequential file).

BACKSPACE instructs MONITOR to backspace 1 physical record on the specified file(s).

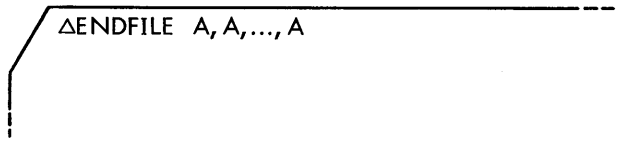
REWIND REWIND pertains to magnetic tape or sequential disc files only.



A is a symbolic name of from 1 to 8 characters.

REWIND instructs MONITOR to rewind the specified file(s).

ENDFILE ENDFILE pertains to magnetic tape or sequential disc files only.

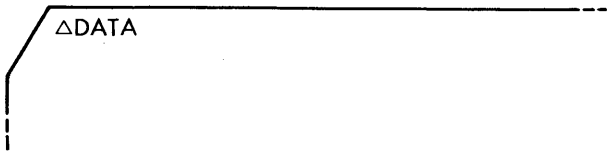


A is a symbolic name of from 1 to 8 characters.

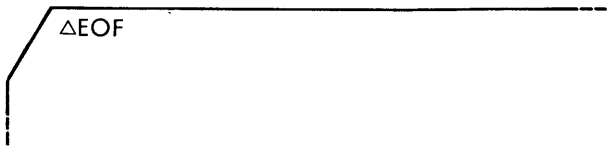
ENDFILE instructs MONITOR to write an end-of-file at the current position of the specified file(s).

INPUT CONTROL MESSAGES

DATA The DATA control card informs the system that there is a data deck to follow. The data deck is for use by the executing program. If a DATA control card appears in the normal sequence of control cards, other than just prior to a data deck, it is ignored.



EOF An EOF card is a terminator for an input source; it separates different types (EI or SI) of input data. If this card appears in the normal sequence of control cards, it is ignored.



Note: An EOF card must follow symbolic input for a META-SYMBOL assembly that involves both symbolic and encoded input. No EOF cards are needed for FORTRAN compilations.

FIN The user places a FIN control card at the end of a stack of jobs, to inform MONITOR that no more information will be received from the C input device. When this card is encountered, all user's operational labels are deleted from the operational label table

At this point, MONITOR types a message to inform the operator that it has completed a stack of jobs and that it requires more information.

OPERATOR CONTROL MESSAGES

After the message field of a △PAUSE control message is output on the system console typewriter, or after the system abort routine (R\ABRT) has been called, one of the following MONITOR messages is typed:

PAUSE TYPE △ABORT; △GO

or

PAUSE TYPE △ABORT; △GO; △RETRY

or

PAUSE TYPE △ABORT; △GO; △RETRY; △ERROR

In response to such a MONITOR message, the operator types one of the following (as specified by the message):

1. △ABORT C/R (C/R = carriage return)
2. △GO C/R
3. △RETRY C/R
4. △ERROR C/R

If △ABORT is typed by the operator, MONITOR then causes the current job or interrupt service routine to be discontinued and one of the following MONITOR messages is typed:

JOB ABORTED

or

INTERRUPT ABORTED

If △GO is typed, the current job or interrupt service routine will continue from the point at which it was halted by the △PAUSE control message.

A △RETRY message causes the appropriate retry routine to be entered, and a △ERROR message causes the appropriate error routine to be entered.

USER-DEFINED CONTROL MESSAGES

The user may define his own control messages, in the following form:

△X...X M,M,...,M

An unrecognized control message will cause a transfer to the program of that name, if such a program is in core at the time that the message is received by MONITOR.

The specification fields (M,M,...,M) of an unrecognized control message can be processed by the user via the MONITOR scan (R\SCAN), symbol table search (R\RSTS), and convert (R\CNVT) routines. (See Section 8.)

SYSTEM RESERVED NAMES

Names reserved for control messages (i.e., △name) must not be used as external definitions. This includes control message names used in debugging.

CONNECT CONTROL MESSAGE

The CONNECT control message is used to associate a particular subroutine with a given interrupt location. (See Section 5.) It has the following form:

△CONNECT (loc, sub(arg_1,...,arg_n))

loc is an octal integer specifying the interrupt location.

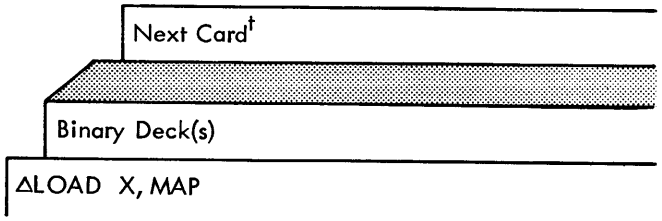
sub is the name of the subroutine to which control is to be given when interrupt loc occurs.

arg_i is the normal argument list used with the subroutine, including any symbolic name that has been defined (i.e., the external definition has been loaded into core). That is, the arg_i may be names of GLOBAL variables or of subprograms.

3. LOADING

BATCH JOBS WITHOUT OVERLAY

The arrangement of a typical program deck for an unsegmented job is shown below. Note that, although no SEG or INCLUDE control messages are needed in this example, an INCLUDE control message may be used to force the loading of a routine from the primary library.

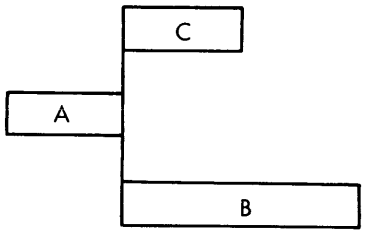


In the above example, the GO file is assumed to contain no binary code; the binary-coded cards are loaded, a load map is printed out on the system console typewriter, and the program is executed.

BATCH JOBS WITH OVERLAY

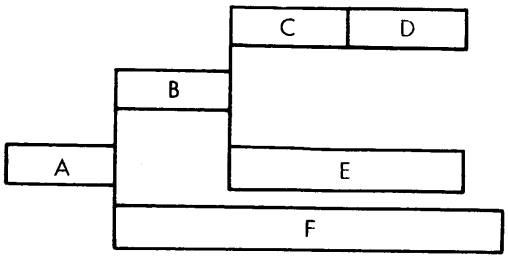
The following examples represent segments diagrammatically as "trees". The horizontal coordinate is used to denote increasing memory allocation and decreasing segment levels from left to right; a vertical coordinate is used to denote overlays.

Example 1: If a program has a main segment labeled A and two overlays, segments B and C, the program could be diagrammed as:



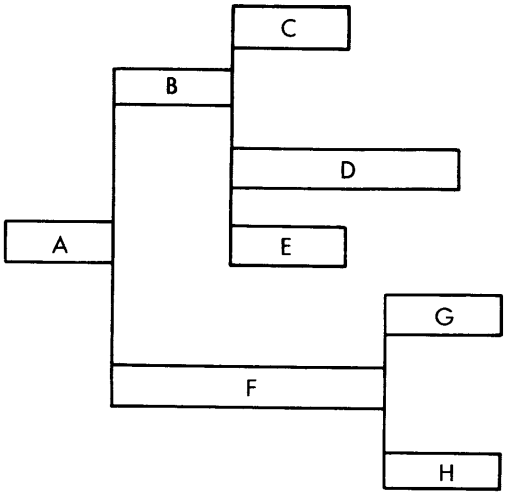
and could be described as △SEG A-(B, C)

Example 2: △SEG A-(B-(C-D, E), F)

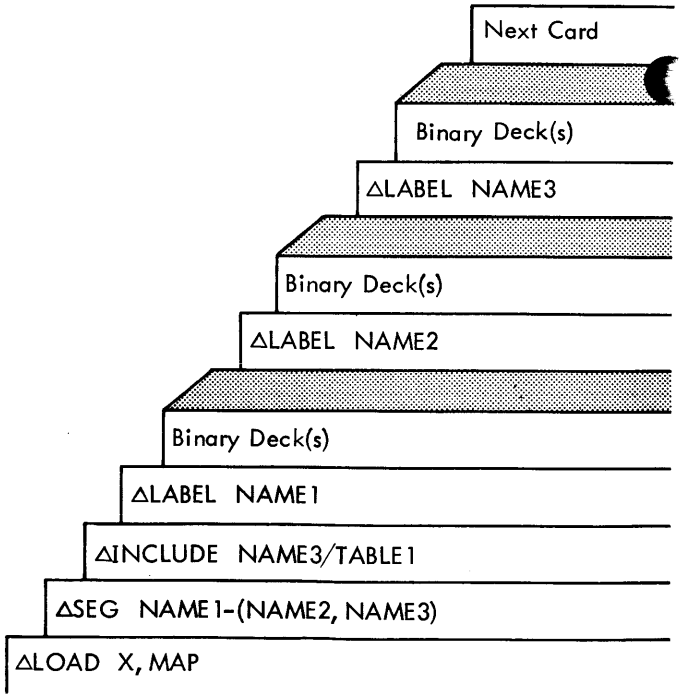


†See Section 4.

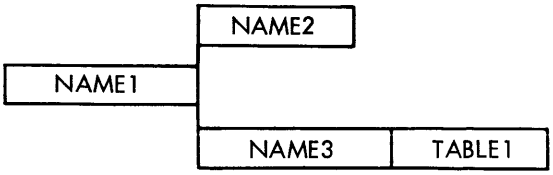
Example 3: △SEG A-(B-(C, D, E), F-(G, H))



The arrangement of a typical program deck for a segmented job is shown below. Note that an INCLUDE control card is used to load a table, from the primary library, with a specified segment.



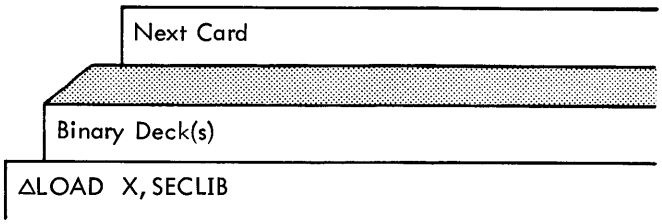
The above example would result in the overlay structure shown in the diagram below.



LOADING TO AND FROM SECONDARY LIBRARY

LOADING TO SECONDARY LIBRARY

The arrangement of a typical program deck to be loaded into the secondary library is shown below. Note that any routine loaded to the secondary library must have at least one external definition.



LOADING FROM SECONDARY LIBRARY

Loading from the secondary library can be accomplished by means of one of the following control messages:

1. △LOAD 'name'
2. △LOAD X, 'name'
3. △LOAD X, 'name', BI

Note that if BI is specified, loading will be from BI, and if the named routine is not found on the BI media, then the secondary library will be searched for the named routine. Any of the LOAD specifications not precluded by use of a 'name' specification may be incorporated in the control message.

LOADING RESIDENT, REAL-TIME PROGRAMS

Typical control messages for loading programs into resident storage for interrupt servicing are as follows:

1. △LOAD X, UPPER
2. △LOAD X, 'name', LOWER
3. △LOAD X, 'name', BI, UPPER

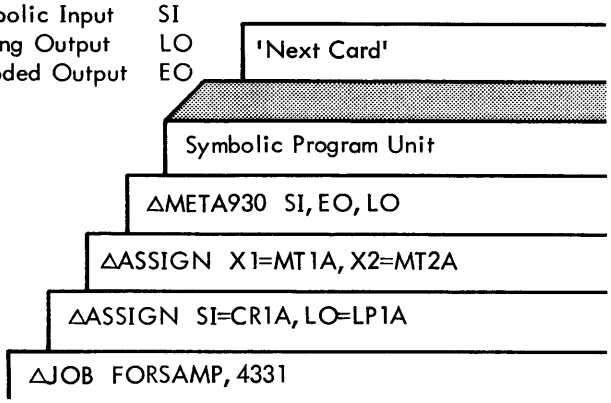
Note that specifying UPPER or LOWER causes the program to be loaded into upper or lower resident core memory, respectively. (See Section 5 for details of real-time operations.)

4. PREPARING THE PROGRAM DECK

The following samples show various ways to prepare program decks for MONITOR operation.

META-SYMBOL INITIAL ASSEMBLY

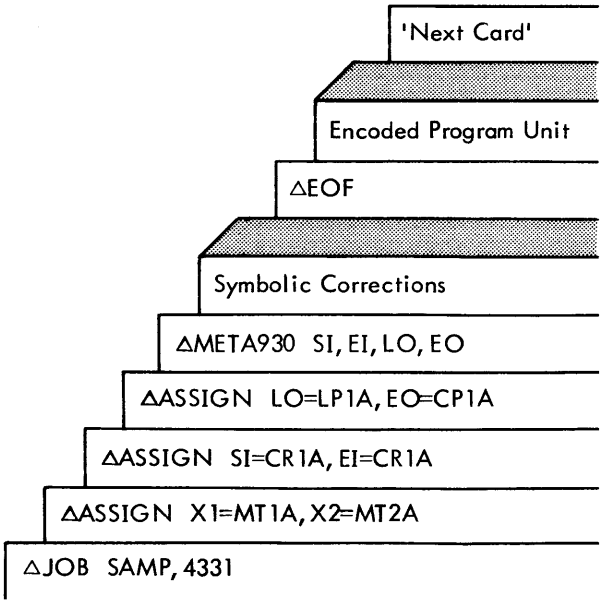
Symbolic Input SI
Listing Output LO
Encoded Output EO



Note: The "next card" must be another control message such as JOB or METAXXXX, etc., denoting the beginning of the next program unit. In all cases of assembly following assembly (namely, EI to EI, EI to SI, SI to SI, or SI to EI), the subsequent source input decks must be preceded by a METAXXXX card.

META-SYMBOL ASSEMBLY WITH CORRECTIONS

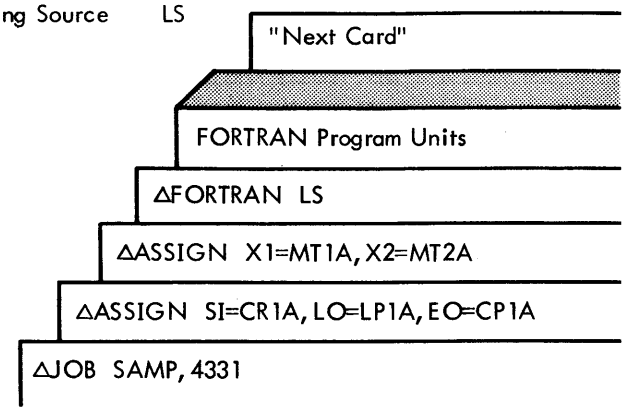
Symbolic Input SI
Encoded Input EI
Listing Output LO
Encoded Output EO



The EOF card separates the two types of program inputs. See the note concerning "next card" in the previous sample.

FORTRAN COMPILATION

(Symbolic Input SI Assumed)
Listing Source LS

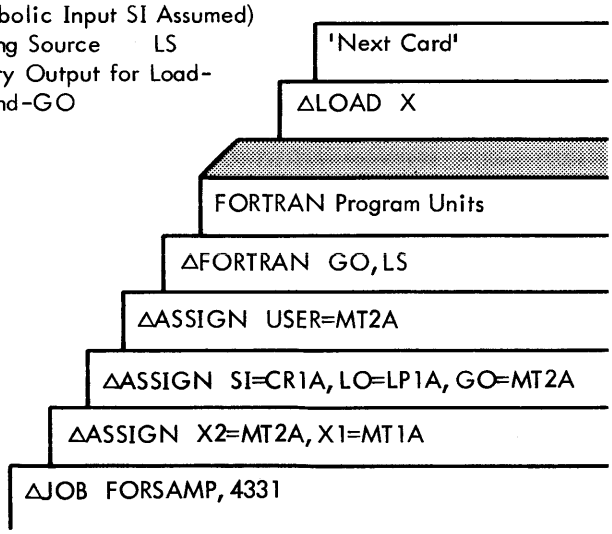


LS is on the LO medium.

The "next card" functions for FORTRAN as it does for META-SYMBOL, except that if it is not a control card, it is assumed to be another FORTRAN symbolic input deck with the same options as specified in the last FORTRAN control message.

FORTRAN COMPILE-AND-EXECUTE

(Symbolic Input SI Assumed)
Listing Source LS
Binary Output for Load-and-GO



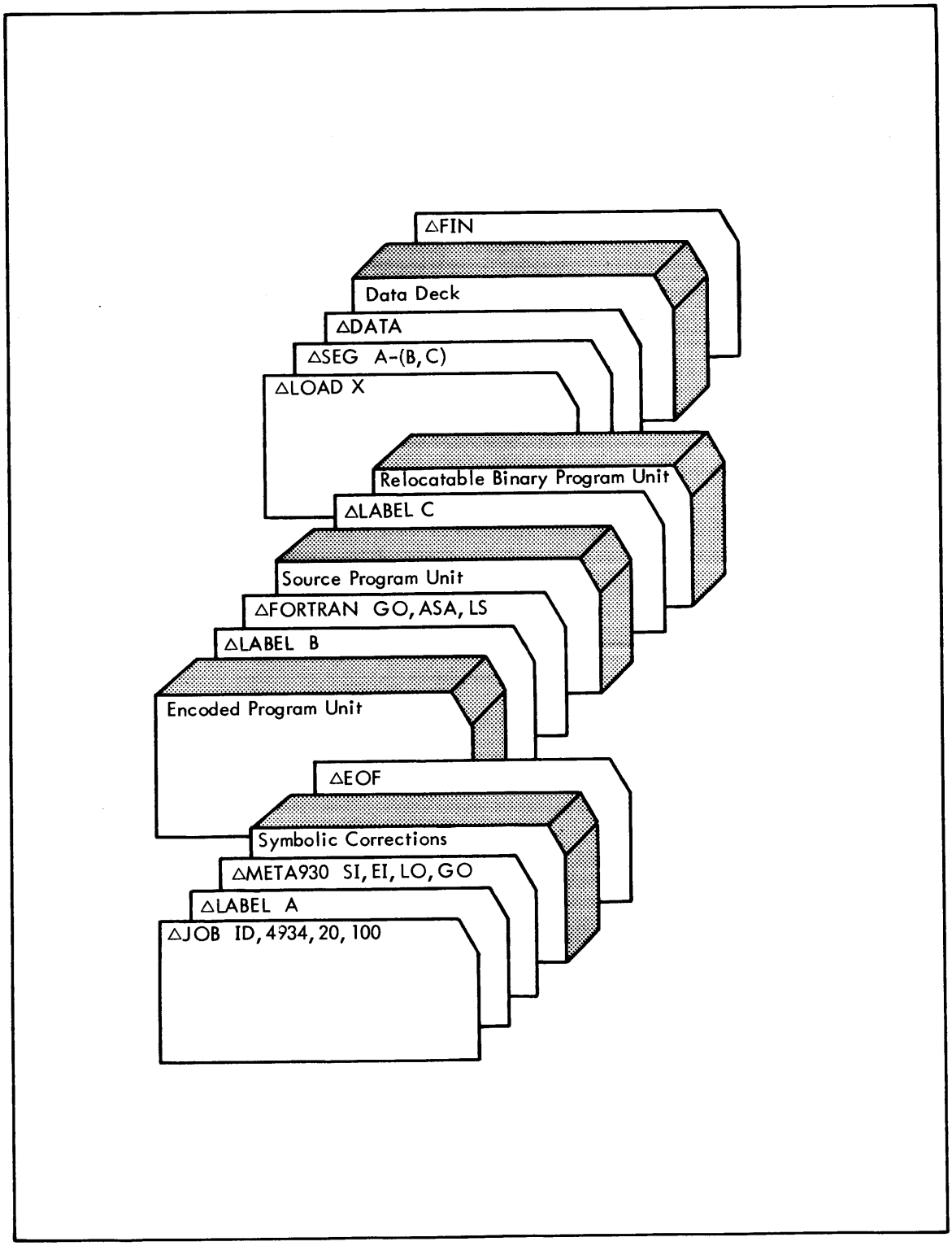
The "next card" must be a △ control card.

The USER assignment assigns and reserves a magnetic tape unit for execution after program compilation. The GO specification on the FORTRAN card informs the FORTRAN processor to generate binary output, for subsequent loading, onto the GO file.

Note that when MONITOR encounters the FORTRAN card, control is transferred to the FORTRAN processor. Therefore, any control card immediately following the FORTRAN card or within the source deck would cause FORTRAN to terminate compilation.

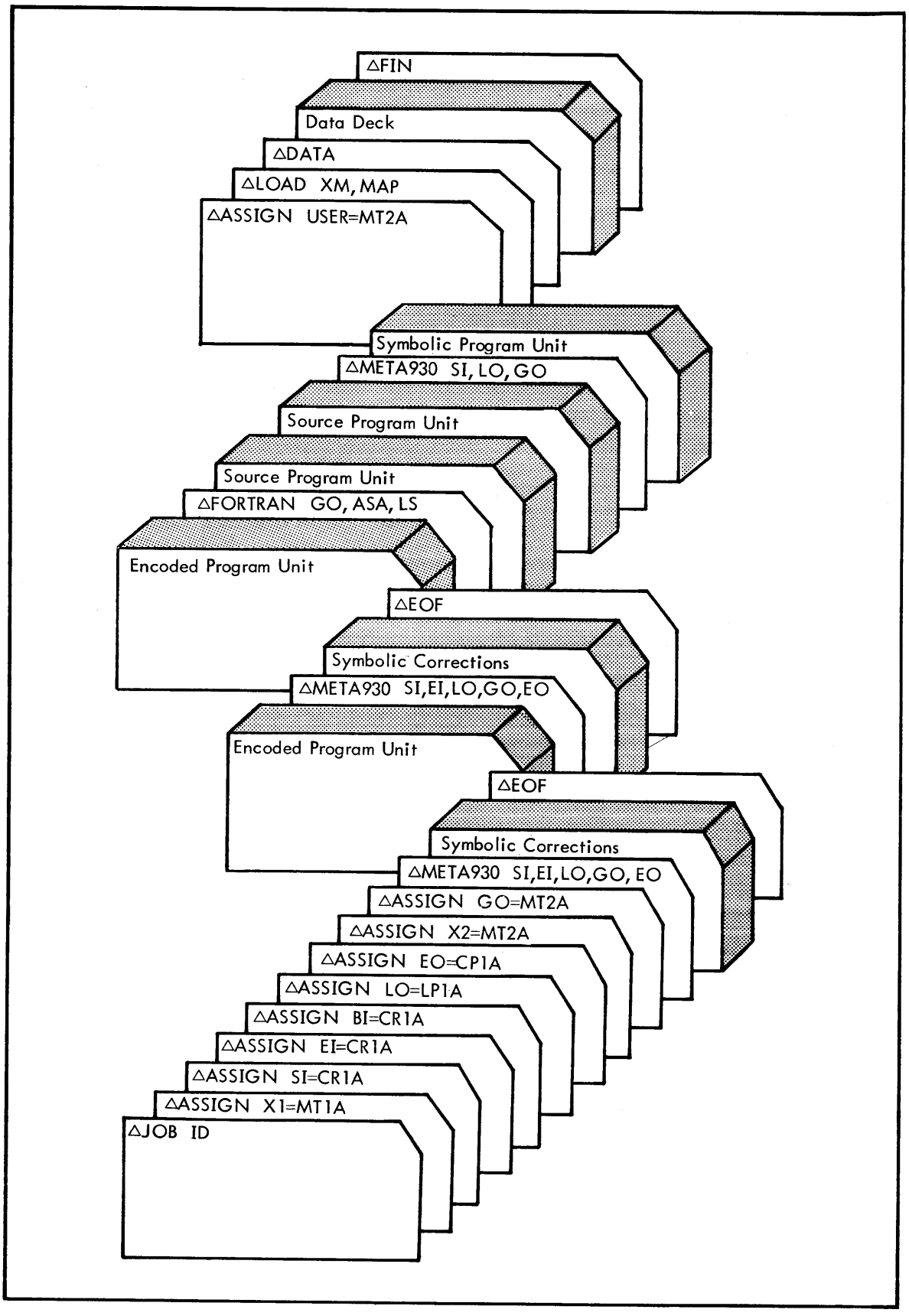
OVERLAY PROGRAM EXAMPLE

An example of an overlay program deck is shown below.



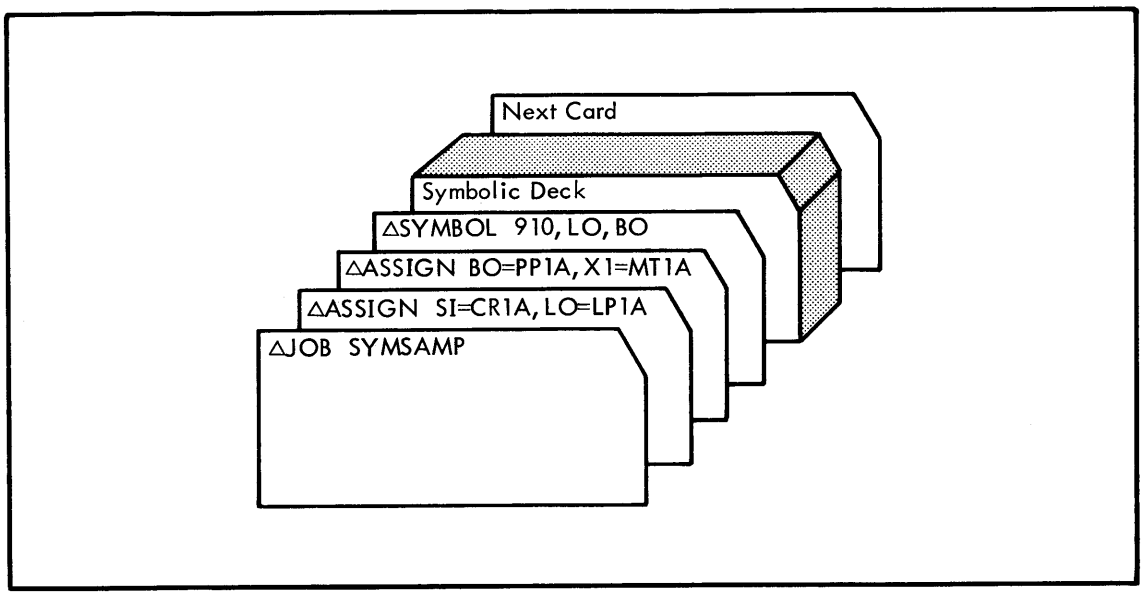
MULTIPLE PROGRAM EXAMPLE

This sample shows some of the many control cards and deck configurations provided in the MONITOR system.



SYMBOL PROGRAM EXAMPLE

An example of a SYMBOL program deck is shown below.



5. REAL-TIME OPERATIONS

MONITOR responds to interrupts occurring during the execution of batch jobs, or during the execution of other real-time service routines of lesser priority, and returns to the interrupted task after completion of the interrupt. Provisions are included for allowing a routine that is responding to an interrupt to call any other routines interrupted while in process, with no danger of losing data. The various interrupt service routines may use a common pool of subroutines.

Real-time programs are loaded in the manner described in Section 3 (see "Loading Resident, Real-Time Programs"). A CONNECT control message (see Section 2) causes code to be generated as shown in the following example (9300 Computer):

△CONNECT (40, SUBR(ARGS))

code executed by the CONNECT message processor

```
        BRM     R\CNCT
        PZE     2
        PZE     040
        PZE     entry
```

code generated in dynamic storage

```
entry   PZE
        DIR     *,2
        BRM     R\SVINTS
        BRM     SUBR    }  calling sequence
        PZE     1       }  for connected
        PZE     ARGS    }  routine
        BRM     R\RSINTS
        BRC     *entry
```

When an interrupt occurs at location 40_8 (in the above example), it results in the execution of a branch to the entry of the calling sequence for the connected routine.

Any interrupt service routine having external references that are not satisfied at load time causes an automatic load of the referenced routines from the secondary library when the instruction referencing the routine is executed.

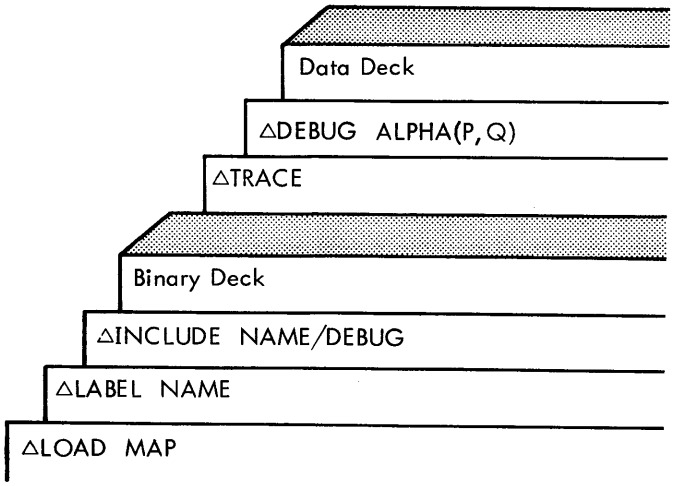
With the exception of I/O operations, any action performed at a given interrupt level will be ceased during the execution of a higher-level interrupt and will be returned to when all higher-level interrupts have been cleared.

A RECURSIVE declaration in FORTRAN defines a reentrant subprogram. Note that this does not mean that such a program may call itself, either directly or indirectly.

Any subprogram that can be entered as the result of more than one interrupt (before its execution has been completed) must be reenterable. Moreover, if a reentrant subprogram has internal subprograms, the internal subprograms must also be reentrant. Note that a protected routine must not initiate any action causing an interrupt to occur during the timespan of its own execution (including I/O operations). Also, a protected subprogram should not contain an internal, reentrant subprogram.

6. DEBUGGING

The debug package is a part of the primary library. It is loaded by using an INCLUDE control message following a LOAD control message. A typical example is shown below.



Upon encountering a DEBUG message (see "Debug Control Message") the Debug program will immediately begin interpretive execution of the specified routine. Only the specified routine and other routines under its control will be executed. All execution will be done interpretively until either (1) the specified routine executes a normal return or (2) a LEAVE message is encountered. In either case, control is returned to MONITOR.

DEBUG CONTROL MESSAGE

The DEBUG control message may specify a complete calling sequence, and has the form

$$\triangle\text{DEBUG} \quad p(a_1, a_2, a_3, \ldots, a_n) \qquad [\text{for } 0 < n \leq 10]$$

where p is the name of the subprogram at which interpretive execution is to begin, and the a_i form the list of arguments required by the subprogram. The a_i may be any externally defined symbol, integer constant, or floating-point constant consisting of a whole number followed by a decimal point and decimal fraction. Both the subprogram name and the argument list are optional, but, if specified, they must be defined (i.e., they must be in core storage). If the subprogram name is not specified, interpretation will begin with the entry to the main program.

DEBUG CONTROL

The following control messages may be used when the debug program is in residence.

TRACE Control Message

This control message is of the form

$$\triangle\text{TRACE} \quad p_1, p_2, p_3, \ldots, p_n \qquad [\text{for } 0 < n \leq 25]$$

where p_i may be

1. the name of a subprogram to be "branch" traced (i.e., printouts for branches only).
2. the name of a subprogram followed by an asterisk, causing the subprogram to be "full" traced (i.e., printouts for all executed instructions).
3. No p_i need be specified, in which case the trace will be a "branch and mark" trace; that is, a trace of all BRM instructions. This trace is always given, regardless of the p_i specified in a △TRACE message.

As many TRACE messages as necessary may be used, with a limit of 25 subprograms named. Also, a breakpoint switch is used to allow an on-line user to cause a "full" trace regardless of what control messages have been read.

TRAP Control Message

This control message is of the form

$$\triangle\text{TRAP} \quad f_1, f_2, f_3, \ldots$$

where each f_i may be either a symbolic instruction or a symbolic address. The effect of a TRAP message is that, every time a specified instruction is used or a specified (effective) address is referenced, a trace-type printout occurs. As many TRAP messages as necessary may be used, with a total maximum of 25 addresses, and a total of 25 instructions. A breakpoint switch may be used by an on-line user to cause a pause following each TRAP printout.

SNAP Control Message

The format of this message is

$$\triangle\text{SNAP} \quad e_1/s_1/n_1, e_2/s_2/n_2, \ldots$$

causing a snapshot dump of n_i words starting at location s_i, whenever the instruction in location e_i is about to be executed. Both e and s are symbolic locations and n is a decimal number.

As many SNAP messages as necessary may be used, with a maximum of 10 snapshot dumps specified.

INSERT Control Message

This message has the form

$$\triangle\text{INSERT} \quad e/w_1, w_2, w_3, \ldots$$

where

1. the instructions specified (the w_i) are to be inserted logically following the instruction at symbolic location e.

2. The w_i may be either octal instructions (8 octal digits), or symbolic instructions of the form

   op loc

   where "op" is a symbolic operation and "loc" is a symbolic or octal location.

As many INSERT messages as necessary may be used, but a maximum of 200 locations are reserved for insertions.

CEASE Control Messages

The format of these messages is

ΔCEASE $m_1/a_1, m_2/a_2, m_3/a_3, \cdots$

where m_i may be any of the previously defined debug message names, and a_i may be any location (or instructions, in the case of TRAP) to which the named function has been assigned.

For example:

```
ΔTRAP     ALPHA+3,STB
ΔTRACE    BETA,GAMMA*,DELTA,ALPHA
ΔSNAP     TW3/RCU/25
ΔINSERT   SST-4/77350031
ΔINSERT   SST+8/LDA B,STA D,LDA C,STA W
  .
  .
  .
ΔCEASE    TRAP/ALPHA+3,SNAP/TW3,INSERT/SST+8
```

The named debug functions are terminated and named insertions are deleted (and the original contents of the affected core locations are restored).

DISPLAY Control Message

This control message allows the immediate display of the specified locations. It has the form

ΔDISPLAY $w_1/n_1, w_2/n_2, w_3/n_3, \cdots$

where the w_i define the symbolic locations at which the displays are to begin, and the n_i indicate the decimal number of words that are to be displayed.

SYMTAB Control Message

This message causes a dump of the resident symbol table. It has the form

ΔSYMTAB

RENTAB Control Message

This message causes a dump of the reentrance chain. It has the form

ΔRENTAB

LEAVE Control Message

This message causes the debug program to be exited and a return to MONITOR to be executed. It has the form

ΔLEAVE

SYSTEM POSTMORTEM DUMP PROGRAM

An entry is provided in the executive to allow the user to obtain a postmortem dump of memory. The dump program is maintained on disc as a separate processor and is called with arguments specifying the extent and type of dump desired.

When the dump program is called, it writes a section of memory onto a disc file and loads itself into the vacated area. All registers are saved and the dump is written on the LO device.

Access to the dump program is via a DUMP message. Arguments may be provided to specify the first and last locations of an area to be dumped. If the required arguments are not provided, all of memory will be dumped. The DUMP message has the form

ΔDUMP START,END

Where START is the first location and END is the last location to be dumped.

7. OPERATOR ACTIONS

MONITOR communicates its needs for operator actions via the system console typewriter. Operator responses to the PAUSE message and calls to the system pause routine are discussed in "Operator Control Messages", Section 2. The calling sequences for the pause (R\PAWS), abort (R\ABRT), and system exit (R\EXIT) routines are given in Section 8.

To initiate a key-in on the 9300 Computer, the operator first presses console button 32 (this is not applicable to 900 Series Computers). To start the key-in message on either 900 Series or 9300 Computers, the operator must type a "Δ" as the first character. MONITOR responds by doing a carriage return and typing a second "Δ". The operator then keys in the remainder of the message and terminates by means of a carriage return.

8. MONITOR INTERFACES

I/O OPERATIONS

To perform an I/O operation, MONITOR must know the description of the basic EOM as well as other pertinent data. The File Description Table (FDT), an area provided by the user in his program, contains this information needed by MONITOR. The arrangement of an FDT is shown below:

File Description Table

Word 1	File status; end address + 1
2	Origin of record
3	Maximum word count
4	Device control EOM
5	I/O format control
6	Address of File Control Block
7	Address of end-action routine

word 1, bit 0 — Set = 1 if IOPS is using this FDT.

bit 1 — Set = 1 if an error occurred during the operation.

bit 2 — Set = 1 if an end-of-file was detected.

bit 3 — Set = 1 if an end-of-tape or end-of-disc-file was detected.

bit 4 — Set = 1 if a beginning-of-tape was detected.

bit 5 — Set = 1 if an attempt to write on a file-protected tape occurred.

bit 6 — Set = 1 if I/O request cannot be honored.

bits 7-8 — Not used.

bits 10-23[†] — Contain the address + 1 of the last word transmitted.

word 2, bits 10-23[†] — Contain the address of the origin of the record to be transmitted.

word 3, bit 0 — If set 1 by calling routine, IOPS will return with bit 6 in word 1 set to 1 whenever I/O request cannot be honored.

bits 10-23[†] — Contain the maximum number of words to be transmitted.

word 4, bits 0-12 — Not used.

bit 13 — Set = 1, start with leader (paper tape only).

Set = 0, do not start with leader.

bit 14 — Set = 1, transmit in binary mode.

Set = 0, transmit in BCD mode.

bits 15-16 — Specify number of characters/word:

Set = 0, 1 char/word.

Set = 1, 2 char/word.

Set = 2, 3 char/word.

Set = 3, 4 char/word.

word 5 may contain any of the following, depending on the device and operation:

a. For magnetic tape spacing operations, this word will contain the number of records to be spaced. The number must be positive if forward, and negative if backward spacing is desired.

b. For typewriter and paper tape operations, a stop character must be specified in bits 18-23.

c. For random-access disc storage operations, a relative sector address must be specified in bits 13-23.

d. For printer operations, a carriage control character may be specified in bits 18-23 (see opcodes 60 and 61).

e. For scanning, must contain the appropriate 4 characters.

word 6, bits 10-23[†] — Contain the address of the FCB for the file being used (i.e., a reference to the FCB which will be satisfied at load time).

word 7, bits 10-23[†] — Contain the address of the user's end-action routine, if required (if no end-action is required, this word must be zero)

The File Control Block (FCB) referenced in word 6 of an FDT has the arrangement shown below. An FCB for any device except discs and drums consists of words 1 and 2 only. Six words are required for disc files.

File Control Block

Word 1	Channel and unit designation
2	Flags and driver pointer
3	First sector address
4	Last sector address
5	Current sector address
6	Current record address

[†]Bits 9-23 for SDS 9300 Computer.

word 1,	bits 1, 6, 17	Channel designation.
	bits 19-23	Unit designation.
word 2,	bit 0	Not used.
	bit 1	Set = 1 if device may be accessed by batch programs.
	bit 2	Set = 1 if device is reserved for real-time programs.
	bit 3	Set = 1 if last use of device was by batch programs. Set = 0 if last use of device was by real-time program.
	bit 4	Set = 1 if last operation on device was a rewind.
	bit 5	Set = 1 if device is unbuffered printer or a Model 9158 punch unit.
	bit 6	Set = 1 if device is random access disc file. Set = 0 if device is sequential disc file.
	bits 7-18	Not used.
	bits 19-23	Index value (pointer) of subroutine entry for this device in I/O table.
word 3,	bits 9-23	First sector address. } file area
word 4,	bits 9-23	Last sector address. } file area
word 5,	bits 9-23	Current sector address. } pointers
word 6,	bits 17-23	Current record address†. } pointers

Note that the format of the first FCB word is identical to that for records on magnetic tape and other units.

I/O MONITORING

The I/O processor, R\IOPS, is a reentrant program having a section of its temporary storage block reserved for each channel connected to the system. When R\IOPS is entered, the channel number is computed. A channel-active test is then made to determine if I/O operations may continue for the current request. When the required channel is not busy, general data (such as unit number, POT word, and the second EOM instruction) are calculated and saved in the temp block for that channel. The type of operation and device are determined, and a branch is made to the appropriate I/O device subroutine.

The device subroutine determines whether or not the device is ready, and sets up the basic I/O instruction that will be used for the operation. All such instructions are stored in the R\IOPS temp block for the required channel. The device subroutine issues instructions that start the interlaced operation, and an interrupt subroutine is connected to the I/O interrupt submonitor to process the termination interrupt.

When the transmission is terminated, the I/O interrupt subroutine is entered and a check for error conditions is made. The interrupt subroutine is device-oriented and may cause some I/O operations to be done. When the requested I/O has been completed and all error flags have been set as required, the interrupt subroutine returns to the I/O interrupt submonitor which then enters the user's end-action routine, if specified. Upon return from the end-action routine, the operation is completed and control is returned to the point of interrupt.

The I/O processor monitors all input from the device assigned as the control message (C) file, and intercepts all △ control messages. When such a message occurs in this file, the I/O routines set the end-of-file indicator in the user's FDT and return. If another attempt is made to read this, the job is aborted.

Output data written on the listing-output (LO) file is also monitored. A line count is kept and the operation code is examined for each call on LO.

Page ejects on the LO device occur at the maximum line count; and title line, data, and page numbers are output, followed by the user's line. This service is not done for any file other than the LO file, evan though other files may be assigned to the same device as LO.

I/O PROGRAMMING

MONITOR uses its I/O processor to perform input/output operations. Via the I/O processor, MONITOR can perform one operation for each unit used by the user's program. Operations for different channels run simultaneously; operations for the same channel run in the order requested. With each input/output operation there is an associated FDT. This FDT is set active as the input/output operation is requested, and is reset to inactive when the operation is completed.

The following devices are serviced by the I/O processor:

1. Card reader/punch
2. Paper tape reader/punch
3. Magnetic tape
4. Line printer
5. RAD File
6. Typewriter

The I/O processor contains a single entry point for I/O operations. Linkage with this entry is via a standard calling sequence with one argument word. The calling sequence is:

```
BRM   R\IOPS
PZE   1
OP    LFDT
 .
 .
 .
```

†Relative to the beginning of the sector block.

where OP is one of the following octal operation codes in bits 3-8:

OP = 0n	Read one record. For magnetic tape or RAD File operations, n (input mode specification) is disregarded. For paper tape and typewriter operations, n may be a value from 0 through 7.
OP = 20	Scan forward, to record identifier specified in word 5 of FDT.
OP = 21	Scan backward to record identifier specified in word 5 of FDT.
OP = 30	Space i records (where i, the record count, is specified in word 5 of the FDT). Applicable to magnetic tape or sequential RAD File.
OP = 31	Write end-of-file. Applicable to magnetic tape or sequential RAD File.
OP = 32	Rewind. Applicable to magnetic tape or sequential RAD File.
OP = 33	Write end-of-file and rewind. Applicable to magnetic tape or sequential RAD File.
OP = 4n	Write one record. For magnetic tape or RAD File operations, n is disregarded. For paper tape and typewriter operations, n (mode) may be either 0 or 1.
OP = 5n	Write one record. This is a printer operation that specifies a skip to channel n before printing.
OP = 60	Write one record. This is a printer operation specifying that the carriage control character for this record is in the fifth word of the FDT.
OP = 61	Write one record. This is a printer operation specifying that the carriage control character for this record is the first character of the line to be printed. The carriage control character is replaced by 060 (blank) before the line is printed.
OP = 7n	Write one record. This is a printer operation specifying that n lines (maximum of 7) are to be upspaced before printing.

TYPEWRITER/PAPER TAPE OPERATION CODES

The following opcodes are used to specify I/O modes for paper tape and typewriter operations in a user's program.

OP is 00

This specifies a BCD input mode in which transmission is immediately terminated on encountering a stop code, exhausting the word count, or encountering a gap on paper tape.

OP is 02

This specifies a BCD input mode in which transmission is immediately terminated only on encountering a stop code or exhausting the word count. If a gap is encountered on paper tape, it is ignored; the tape is spaced past the blank area and transmission resumes.

OP is 04

This specifies a BCD input mode in which transmission is immediately terminated on encountering a stop code, exhausting the word count, or encountering a gap on paper tape. If the maximum word count is exhausted, paper tape is spaced to the next gap.

OP is 06

This specifies a BCD input mode in which transmission is immediately terminated on encountering a stop code. Any gaps on paper tape, encountered before the word count is exhausted, are ignored. If the maximum word count is exhausted, paper tape is spaced to the next gap.

OP is 01

This specifies a binary input mode in which transmission is immediately terminated on exhausting the word count or encountering a gap on paper tape.

OP is 03

This specifies a binary input mode in which transmission is immediately terminated on exhausting the word count. If a gap is encountered on paper tape, it is ignored; the tape is spaced past the blank area and transmission resumes.

OP is 05

This specifies a binary input mode in which transmission is immediately terminated on encountering a gap on paper tape or exhausting the word count. If the maximum word count is exhausted, paper tape is spaced to the next gap.

OP is 07

This specifies a binary input mode. Any gaps on paper tape, encountered before the word count is exhausted, are ignored. If the maximum word count is exhausted, paper tape is spaced to the next gap.

OP is 40

This specifies a BCD output mode in which transmission is immediately terminated on encountering a stop code or exhausting the word count.

OP is 41

This specifies a binary output mode in which transmission is immediately terminated when the word count is exhausted.

Table of I/O Modes

The options implemented in the various paper tape and typewriter I/O modes are summarized in the following table.

Paper Tape and Typewriter I/O Modes

Octal OP	Mode	I/O	Pad?	Delete? (077)	Change This	to This	Stop Code?	Move to Gap	Ignore Gap
00	BCD	I	060	Yes	012	060	Yes	No	No
01	Bin.	I	00	No	--	--	No	No	No
02	BCD	I	060	Yes	012	060	Yes	No	Yes
03	Bin.	I	00	No	--	--	No	No	Yes
04	BCD	I	060	Yes	012	060	Yes	Yes	No
05	Bin.	I	00	No	--	--	No	Yes	No
06	BCD	I	060	Yes	012	060	Yes	Yes	Yes
07	Bin.	I	00	No	--	--	No	Yes	Yes
40	BCD	O	No	No	060	012	Yes	No	No
41	Bin.	O	No	No	--	--	No	No	No

MAGNETIC TAPE OPERATIONS

READING

A record from the magnetic tape specified in the FCB referred to in the sixth word of the FDT is read into memory. The starting address is specified in the second word of the FDT. Reading continues until the end-of-record is reached or until the word count (Word3, FDT) is reduced to zero. In both cases, the tape is positioned in the gap following the record read. If an error has occurred, the error flag bit in the FDT status word will be set to 1. The end-of-file, beginning-of-tape, and end-of-tape error flags in the FDT status word are set if those conditions are encountered.

WRITING

Before each write operation is attempted, the tape is tested for file-protect. If a file-protect is detected, the file-protect bit in the FDT status word is set and control is returned to the user's program.

One physical record, of the length specified by the word count in the FDT, is written. If an error occurs, and the write is retried and is still unsuccessful, the error flag in the FDT status word is set. The tape stops after the last write attempt.

If the end-of-tape indicator is set during the write operation, the end-of-tape flag in the FDT status word is set.

SPACING

Spacing is accomplished by using the tape scan operation in the 4-characters-per-word mode. Spacing may be either forward or backward, as specified in the fifth word of the FDT. Spacing is terminated by an end-of-tape, end-of-file or beginning-of-tape signal.

SCANNING

The specified file will be scanned for a record identifier identical to the key word specified in word 5 of the FDT. For a forward scan, the identifier will be the last 4 characters of each record. If the operation is a reverse scan, the identifier will be the first 4 characters of each record, in reverse order.

If an end-of-file, beginning-of-tape, or end-of-tape condition occurs, the scan will terminate and the appropriate flag will be set in the FDT.

WRITE END-OF-FILE

The tape is tested for file-protect. If a file-protect is detected, the file-protect bit in the FDT status word is set, otherwise, an end-of-file is written. If an error occurs, the tape is repositioned and the operation retried.

CARD READER/PUNCH OPERATIONS

READING

One record is read from the device specified in the FDT. If the number of words specified in the table is less than the number of words in the record, the remaining words are lost. If an error occurs or an end-of-file condition is detected, appropriate flags are set in the status word of the FDT.

PUNCHING

One record is punched on the device specified in the FDT. If an error occurs, an error flag is set in the status word of the FDT.

ERRORS

If a feed check error has occurred, the ending address in word 1 of the FDT will be equal to the starting address. If a validity check error has occurred, the ending address in word 1 of the FDT will not be equal to the starting address. This difference allows the program to determine which type of error has occurred.

LINE PRINTER OPERATIONS

PRINTING

One record is printed on the line printer specified by the FDT. The format control is dependent on the value of the OP in the user's calling sequence. If the word count in the FDT specifies more than 132 characters, only the first 132 will be printed. If the word count specifies zero words, the format control character will be interpreted, the required format action taken, and control returned to the user.

ERRORS

A channel error will cause the error bit to be set in the status word of the FDT.

CARRIAGE CONTROL

The following tables list the carriage control characters which R\IOPS will accept. Table 1 gives the standard SDS format control characters which are specified in the FDT or in the first character position of the output record. If one of these characters is used, the mode of the I/O request (FDT word 4, bit 14) must be binary (i.e., FDT word 4, bit 14 = 1). If the mode of the I/O request is BCD, the carriage control character will be assumed to be a FORTRAN character, as described in Table 2.

Table 1. Carriage Format Control Characters

00	0	Skip to format channel 0
01	1	Skip to format channel 1
02	2	Skip to format channel 2
03	3	Skip to format channel 3
04	4	Skip to format channel 4
05	5	Skip to format channel 5
06	6	Skip to format channel 6
07	7	Skip to format channel 7
40	-	Do not upspace before printing
41	J	Upspace 1 line before printing
42	K	Upspace 2 lines before printing
43	L	Upspace 3 lines before printing
44	M	Upspace 4 lines before printing
45	N	Upspace 5 lines before printing
46	O	Upspace 6 lines before printing
47	P	Upspace 7 lines before printing
other		Upspace 1 line before printing

Table 2. FORTRAN Carriage Format Control Character

00	0	Double space before printing
01	1	Skip to top of form
20	+	Do not upspace before printing
other		Skip one line before printing

PAPER TAPE AND TYPEWRITER OPERATIONS

Paper tape and typewriter operations are performed in either the BCD or binary mode as described below. These modes specify the type of character testing to be performed by the I/O processor during data transmission. The user's program may also specify the termination conditions for the operation, by placing a stop-character code in word 5 of the FDT, prior to calling the I/O processor.

The modes of operation, BCD and binary, are not hardware modes of operation, but only convenient names describing the options available (specified by I/O opcodes).

BCD MODE

Delete Character

A BCD input operation will ignore any 077 (delete) codes read.

Blank Replacement

Either a 060 or a 012 will produce a blank space on a line printer. However, a typewriter will type a 060 code as a ♭ and a 012 code as a space. In a BCD output operation, all 060 codes are converted to 012 codes. In a BCD input operation, all 012 codes are converted to 060 codes.

Padding of Partial Words

In a BCD input operation for which the number of characters read is not an integral multiple of the number of characters per word (specified in word 4 of the FDT), the last word is padded with trailing 060 codes. No equivalent option is provided for output operations, since output terminates on the transmission of a stop character.

Stop Character

In a BCD mode, encountering a character equal to the stop code causes transmission to be terminated when the processing of that character is completed.

BINARY MODE

Delete Character

In a binary mode, the transmission of 077 code is handled in precisely the same way as any other code.

Blank Replacement

In a binary mode of transmission, no character replacement is done.

Padding of Partial Words

In a binary input operation for which the number of characters read is not an integral multiple of the number of characters per word, the last word is padded with trailing zeros. (No equivalent option is provided for output operations.)

Stop Character

In a binary mode of transmission, no stop character options exist, as no stop code is used.

DISC FILE OPERATIONS

RANDOM-ACCESS OPERATIONS

Reading

Data can be read from disc storage files by specifying a sector address in word 5 of the FDT. The file used must have been assigned to disc storage by an ASSIGN message and must have had a maximum file size specified. (When the file was assigned to disc storage, file sectors were allocated for it by MONITOR and its disc address was saved in the FCB.) The sector address specified in the FDT is treated as a relative address (i.e., relative to the beginning of the file). Words are read from the file, beginning at the specified sector and continuing until the word count is reduced to zero. A maximum word count of 4096 may be specified in word 3 of the FDT.

Writing

Data can be written on disc storage files by using the word count and sector address specified in the FDT. The word count is checked to ensure that the data to be written does not exceed the limits of the file specified.

Errors

If a channel error occurs, the error bit of the status word of the FDT is set to 1.

If an attempt is made to transmit data beyond the limits of the specified file, only data within the specified file is transmitted and the end-of-disc-file bit in the status word of the FDT is set to 1.

SEQUENTIAL OPERATIONS

Files assigned to disc storage may be treated in much the same manner as sequential magnetic tape files. That is, they can be manipulated with the same I/O commands as magnetic tape files. Each disc operation is similar to a corresponding magnetic tape operation. Sequential disc files are manipulated by the disc I/O subroutine by arranging them into chained sector blocks and maintaining these blocks by the use of pointers. The pointers for each file are stored in the FCB.

As data is written on a file and new blocks of sectors are needed to contain it, new blocks are obtained from a disc sector map and are added to the chain. Within each sector block are record control words that depict the individual records written on the file in that block. These record control words are used by the disc I/O subroutine in manipulating and maintaining disc files.

When a file is released during the processing of a job (not at the end of a job), the sectors allocated for the file are placed in an empty-sector pool. If, at some later time, no other sectors are available the empty-sector pool is purged and its sector blocks are reallocated. The empty-sector pool is always purged at the end of a job.

The format of the sector block record control words is given below.

Sector Block Record Control Words

word 1	bits 9-23	Previous sector address	sector control words
word 2	bits 9-23	Next sector address	
word 3	bits 0-11	Previous record length	logical-record control word
	bits 12-23	Current record length = n	
word 4	bits 0-23	First word of record	
3+n	bits 0-23	nth word of record	
3+n+1[†]	bits 0-11	Previous record length = n	logical-record control word
	bits 12-23	Current record length = m	

DISC STORAGE SECTOR MAP

A disc storage sector map is maintained in core memory, for use in allocating disc storage. The map contains a bit for each sector block on disc. The sector map is updated in core memory each time that a sector block is allocated. The map is also maintained and updated on disc by the assign processor for each file reserved by an ASSIGN control message. On completion of a batch job, the sector map in core memory is initialized by overlaying it with the disc map.

In the following description of map searching, a sector block is referred to as a sector.

DISC SECTOR MAP SEARCH

The disc sector map search routine performs the function of searching the disc sector map in core, obtaining the address of the available sectors, and allocating these sectors by setting map bits. There are two entry points to the routine.

Entry point R\SECT is used when a single sector block (sectors are always allocated in a block of 4 sectors) is to be allocated. The calling sequence for R\SECT is:

```
BRM   R\SECT
PZE   0
```

No arguments are specified and the address of the allocated sector is returned in the A register.

[†]Records may span sector blocks, so that the logical record control word is not always the first word after the sector control words.

The second entry point, R\SCTS, is used when a block of sectors is to be allocated. The calling sequence for R\SCTS is:

```
BRM   R\SCTS
PZE   3
PZE   SIZE
PZE   FSTSCT
PZE   LSTSCT
```

where:

SIZE — is a cell containing the size, in words, of the block of storage

FSTSCT — is a cell containing the disc address of the first sector in the allocated block.

LSTSCT — is a cell containing the disc address of the last sector in the allocated block.

Error Conditions

If the search program is unable to locate any available sectors, a test is made of the empty-sector pool, which contains all sectors that have been allocated and released during a job. If there are sectors in the pool, they are returned to the sector map and reallocated. If the empty-sector pool does not contain any sectors, the following options apply:

1. If current sectors were to be allocated to a batch job, the job is aborted and a message typed to inform the operator.
2. If a real-time program is operating and a batch job is in memory (not swapped out), the batch job is aborted and a message typed. When a batch job is aborted during real-time processing, the batch symbol table is searched for the FCBs and the sectors allocated for those files are released.
3. If a real-time program is operating and a batch job does not exist in memory, the current interrupt level is cleared and the previous interrupt program continued. A message is typed describing the reason for clearing the interrupt and the interrupt level.

MONITOR SUBROUTINES

MONITOR subroutines of general interest are discussed here, as an aid in understanding the basic control functions performed by MONITOR. Note that the user does not normally participate in the performance of such functions, aside from providing the necessary control messages (via control cards or typewriter).

SYSTEM PAUSE ROUTINE

The calling sequence for the pause routine is as follows:

```
      BRM   R\PAWS
      PZE   n          For 1 ≤ n ≤ 3
      PZE   MSG        (required)
      PZE   RETRY      (optional)
      PZE   ERROR      (optional)
      .
      .
      .
MSG   PZE   size       Message length, in words
      TEXT  m, message m = size*4
```

Note that RETRY and ERROR are optional return points. If the ERROR exit is desired, RETRY must also be specified (the RETRY exit could be a dummy). An unconditional branch to the optional exits is executed when the exit is taken.

Routine R\PAWS types the specified message and permissible operator actions according to the exits defined by the call. The typeout is in the following format:

PAUSE TYPE ΔABORT; ΔGO; ΔRETRY; ΔERROR

Note that if the ERROR (or RETRY and ERROR) exit was not defined in the call to R\PAWS, the ΔERROR (or the ΔRETRY, ΔERROR) option would not appear in the typeout.

SYSTEM ABORT ROUTINE

The calling sequence for the abort routine is as follows:

```
      BRM   R\ABRT
      PZE   1
      PZE   MSG
      .
      .
      .
MSG   PZE   size       Message length, in words
      TEXT  m, message m = size*4
```

If R\ABRT is called from a batch job, the routine will cause the following message to be output:

JOB ABORTED

However, if the abort routine is called from a resident real-time routine, the message

INTERRUPT ABORTED

will be output and MONITOR will return to the next-lower active interrupt level (including the "zero" level, if no interrupts are active).

SYSTEM EXIT ROUTINE

The calling sequence for the system exit routine is as follows:

```
BRM   R\EXIT
PZE   0
```

This routine is the normal exit from a batch job, and may not be called by a resident real-time program.

CONTROL MESSAGE SCAN ROUTINE

The control message scan routine, R\SCAN, is used to scan the specification fields of a control message. It returns each field delimiter and the character string preceding the delimiter, as well as the transfer address associated with the delimiter. The calling sequence for R\SCAN is on the following page.

```
                BRM     R\SCAN
                PZE     1
                PZE     TABLE
                :
DELIM           FORM    6,18
TABLE           PZE     maxchar
                PZE     n
                DELIM   'd1',addr1
                DELIM   'd2',addr2
                :
                DELIM   'dn',addrn
NO.CHAR         PZE     0
STRING          RES     m
```

where

maxchar = The maximum number of characters allowable in the string. (i.e., 4*m)

n = The number of delimiter entries.

d_i = Delimiter character.

$addr_i$ = The transfer address associated with the delimiter.

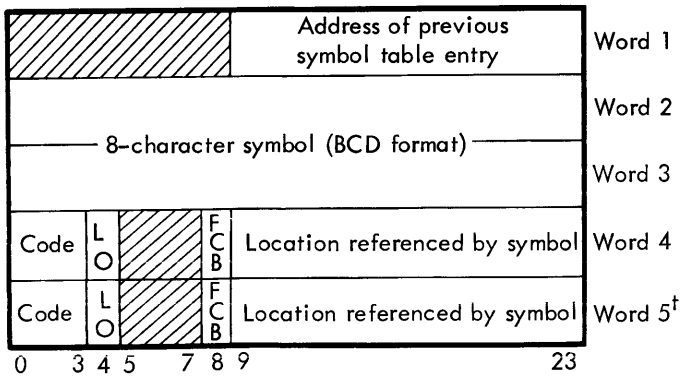
R\SCAN will return the field delimiter (one of the set specified in the delimiter table) in bits 0 - 5 of the A register, the associated transfer address in the index register (900 Series; X1 on a 9300), the number of characters in the string in NO.CHAR, and the character string itself, left-justified with trailing blanks, in STRING.

If a delimiter of the specified set cannot be found in the control message, R\SCAN will set bit 0 of NO.CHAR and will reset bits 1 - 23, the A register, and the index register. If the character string length exceeds maxchar, but a delimiter is found, R\SCAN will return with normal settings, except that bit 0 of NO.CHAR will be set.

RESIDENT SYMBOL TABLE SEARCH ROUTINE

A call on R\RSTS causes the resident symbol table to be searched for the specified symbol. The resident symbol table is composed of external definitions of all programs currently in memory and is broken into two parts; (1) MONITOR and batch program external references, and (2) MONITOR and resident program external references. Consequently, except for symbolic file references, symbols used by batch and resident programs will not conflict.

Each symbol table entry has the following format:



†Optional; see code 12 below.

Code (octal) — Reference

00 = Unused.

01 = Unused.

02 = Reference to a non-disc (or drum)FCB.

03 = Reference to a system file FCB. A system file cannot be reassigned, nor can another file be assigned to it.

04-07 = Unused.

10 = External definition reference.

11 = Unused.

12 = Reference to a disc (or drum) FCB. This symbol table entry will consist of 5 words. If this file is reassigned, word 4 will be saved in word 5, and the new assignment will be reflected in word 4.

13 = Same as code 03, but this file will be undefined in the magnetic tape version of the system.

14-17 = Unused.

LO

0 = This is not a reference to the LO file FCB.

1 = This is a reference to the LO file FCB.

FCB

0 = This is not a reference to an FCB.

1 = This is a reference to an FCB.

The calling sequence for the symbol table search routine is

```
                BRM     R\RSTS
                PZE     1
                PZE     STRING
STRING          TEXT    8,symbol
                PZE     K
```

where

K = 1, search batch program symbol table.

2, search resident program symbol table.

3, search both batch and resident program symbol tables.

When R\RSTS finds the symbol defined by STRING, the location defined by the symbol will be returned in the A register (bits 9-23). The location of the symbol table entry will be returned in the index register (900 Series; X1 for 9300). If the symbol cannot be found in the symbol table, bit 0 of the A register will be set.

If K specifies that both symbol tables are to be searched, the resident program symbol table will be searched first.

SYSTEM BCD TO BINARY CONVERSION ROUTINE

This program will convert a BCD string of numbers to a binary value. If the string contains a decimal point, it will be considered to be a single-precision floating-point value. If the string begins with a zero (and contains no decimal point), it will be considered to be an octal value.

The calling sequence for R\CNVT is

```
          BRM     R\CNVT
          PZE     1
          PZE     NO.CHAR
          :
          :
NO.CHAR   PZE     n
STRING    TEXT    m, string
```

where

n = Number of characters to be used.

m = Integer ((n+4-1)/4).

Integer decimal and octal values will be returned in the A register, and floating-point values will be returned in the A and B registers. Values which are not representable, due to excess magnitude, will result in the A and B registers being set to maximum or minimum values (dependent on the sign of the value). Illegal strings will cause the maximum settings.

RESIDENT LOADER

Resident Loader Control Routine

All programs are loaded under the direction of the resident loader control routine, which informs the semiabsolute loader of load locations and library files required and also generates the implicit call linkage table. Resident loader control is entered via the calling sequence

```
BRM  R\LOAD
PZE  system file name
PZE  system program name
PZE  load location, tag
```

system file name — is one of the following system files:

1. R\PROC the processor file
2. R\OVRL the overlay file (includes user's routines)
3. R\PRIL the primary library file
4. R\SECL the secondary library file (includes user's routines)

system program name — is the address of the first of two consecutive words containing the name of a program that is to be loaded and that can be found in the file specified by file name.

load location — is the address of a word containing the location at which the program is to be loaded. The value of tag determines whether or not the program is entered following loading.

tag = 0: the program is not entered and control returns to the caller.

tag = 1: the program is entered at the location specified by program name.

If the program entered returns control to resident loader control, resident loader control returns to its caller.

Semiabsolute Loader

The semiabsolute loader performs all program loading functions for the system. Given the program and file names as arguments, it searches the file dictionary to obtain the program's disc address. It then reads the program directly into the memory location into which it is to be loaded.

The seimabsolute loader calling sequence is

```
BRM  R\SALD
PZE  file name
PZE  program name
PZE  load location
```

file name — contains the name of the file in which the program to be loaded can be found.

program name — is the address of the first of two words which contain the name of the program to be loaded.

load location — contains the address at which the program is to be loaded.

Implicit Call Processor

The implicit call processor is entered as a result of attempting to execute an instruction that referenced an undefined external symbol at load time. When such an unsatisfied reference exists at load time, the resident loader control routine makes an implicit call link in dynamic storage entry. It is through such an entry that the implicit call processor is entered.

Upon entry, the implicit call processor searches the resident symbol table for the required symbol. This is done in case the symbol has been defined by a load subsequent to the link entry being generated. If the symbol table is not in residence, a search is made of a system file, either overlay, processor, or secondary library, depending on the type of program being executed, and the program containing the required symbol is loaded. After loading, the value of the required symbol is merged into the replaced instruction within the implicit call link entry and the instruction is executed interpretively.

The implicit call link entry is

```
REFi   BRM    CALLi     Original reference replaced by
                         this instruction
CALLi  PZE    0         First word of link entry
       BRM    R\IMP     Call to implicit call processor
       TEXT   8, name   8-character name of referenced
                         item
       instruction      instruction originally at REFi
```

Manual Loading

The user may cause a program to be loaded, without execution, by referring to it as the operand of an NOP instruction. An example is given below.

```
       NOP    TABLE3
```

The above example would result in the generation of the following:

```
       BRM    IMPi
       :
       :
IMPi   PZE
       BRM    R\IMP
       TEXT   8,TABLE3
       NOP
```

STANDARD CALLING/RECEIVING SEQUENCES

900 SERIES COMPUTERS

The calling/receiving sequences for the SDS 900 Series use operation code bits to determine the data type of the arguments. These data type codes are derived as follows:

Data Type Codes

Calling or Receiving Sequences				
Type	Normal	Octal	Protected	Octal
Integer	INTG	001	INTP	041
Real	SNGL	002	SNGP	042
Double-precision	DOUB	004	DBLP	044
Complex	CMPX	010	CPXP	050
Logical	LOGL	020	LGLP	060
(Labels, subprogram identifiers, etc.)	PZE	000	PZEP	040

Receiving Sequences Only				
Type	Normal	Octal	Protected	Octal
(Real, integer, or double-precision)	ANY	007	ANYP	047
(ANY plus complex)	ALL	017	ALLP	057
(ALL plus logical)	EVRY	037	EVRP	077
(Variable number of arguments)	VARG	777		

The above octal operation codes are composed of bits set to 1 according to the following conventions:

bit 3 = 1: protected (calling – cannot be stored into; receiving – will store into)

bit 4 = 1: logical

bit 5 = 1: complex

bit 6 = 1: double-precision

bit 7 = 1: real

bit 8 = 1: integer

Calling Sequence. The standard calling sequence is

```
BRM     subprogram
PZE     n
type    arg1
type    arg2
type    arg3
.
.
.
type    argn
```

where n specifies the number of calling arguments, and the arg_i are the addresses of the calling arguments. The "type" operation codes indicate the data type, if any, of the calling arguments.

Receiving Sequences. The standard receiving sequence is

```
entry       PZE     0           Entry point
            BRM     9SETUP      Call to set up n arguments
            PZE     NO. ARGS
            :
            :
            BRR     entry       Exit (RETURN)
            :
            :
NO. ARGS    PZE     n           n = number of receiving arguments
ARG1        type    0           Calling argument
.           .       .           addresses moved
.           .       .           to these locations
.           .       .           by 9SETUP
ARGn        type    0           Local variables,
:           :       :           temps, etc.
:           :       :
```

The above receiving sequence is modified slightly for routines that expect varying numbers of arguments, as in

```
entry       PZE     0           Entry point
            BRM     9SETUP      Call to set up a variable number of arguments
            PZE     NO. ARGS
            :       :
            BRR     entry
            :       :
NO. ARGS    VARG    n           n = number of fixed arguments
ARG1        type    0
ARG2        type    0           Fixed arguments
ARG3        type    0
:           :       :
ARGn        type    0
ARGv        type    *,X1        Variable arguments
            PZE     m           m = number of variable arguments
            :       :
m           PZE     0
```

Receiving Sequences for PROTECTED Routines. All FORTRAN routines designated as PROTECTED have the receiving sequence

```
entry       PZE     0           Entry point
            DIR     0,2
            BRM     R\PROT      Protection routine
            PZE     NO. ARGS
            :
            :
            BRM     R\UNPT      Unprotect routine
            BRR     entry
            :       :
            :       :
```

```
NO. ARGS    type    n          (Either PZE or VARG)
ARG1        type    0
ARG2        type    0
ARG3        type    0
 :           :
 :           :
            etc.
```

Receiving Sequences for Reentrant Routines. FORTRAN routines that have been declared to be RECURSIVE, and reentrant assembly-language programs, have the receiving sequence

```
entry       PZE     0          Entry point
            DIR     0,2
            BRM     R\RENT     Reentrance monitor
            PZE     TEMPS      Temp block
            PZE     NO. ARGS
             :
             :
            BRR     RETURN
             :       :
             :       :
TEMPS       PZE     END-$+1    Size of temp block
LAST        PZE     0
RETURN      PZE     0
NO. ARGS    type    n          (Either PZE or VARG)
ARG1        type    0
ARG2        type    0
ARG3        type    0          Fixed and/or variable arguments,
 :           :                  local variables, and temps
 :           :
END                            (Last temp cell)
```

The CONNECT Statement. The FORTRAN IV CONNECT statement generates code having the same effect as the following:

FORTRAN statement:

```
CONNECT (40, SUB(ARG1, ARG2, ARG3))
```

Generated code:

```
            BRM     R\CNCT     CONNECT routine
            PZE     2
            PZE     040
            PZE     entry
            BRU     NEXT
entry       PZE     0
            BRM     SUB        CONNECTED routine
            PZE     3
            type    ARG1
            type    ARG2
            type    ARG3
            BRR     entry
NEXT         .       .
             :       :
```

9300 Computer

A reference to a subprogram by a source program causes the compiler to generate a calling sequence to the subprogram. Also, some subprograms are called implicitly by the source program. In either case, the referenced subprogram has a receiving sequence that facilitates the exchange of argument addresses between the calling statement and the subprogram. Such sequences are of three types: standard, nonstandard, and special.

Standard Calling Sequence. The standard calling sequence is

```
BRM     subprogram
PZE     n
type    arg1
type    arg2
type    arg3
.
.
.
type    argn
```

n = the number of calling arguments

Standard Receiving Sequence. The standard receiving sequence is

```
entry   PZE     0
        BRM     9SETUPN
        PZE     n
        type    0
        type    0
        .
        .
        .
```

n = the number of receiving arguments

Standard Receiving Sequence with Conversion. It is often desirable to allow a subprogram to accept an argument of any of several types (e.g., any of the numeric, but not logical, forms). However, because it is practical to write the subprogram to process only one type, a procedure is needed to perform the desired conversion. An example is given below.

Calling, for N = 2:

```
BRM     SPROG
PZE     2
SNGL    arg1
LOGL    arg2
```

Receiving:

```
SPROG   PZE     0
        BRM     9SETUPNC
        PZE     2
        ANY     0
        LOGL    0
        DOUB    1TEMP
        LOGL    2TEMP
```

The PZE 2 defines two receiving arguments and two conversion items

The last two arguments of the above receiving sequence specify the types to which the corresponding calling arguments are to be converted and the locations into which the converted arguments are to be stored.

Standard Receiving Sequence for a Variable Number of Arguments. When it is desirable to write a subroutine capable of handling a variable number of arguments, the receiving sequence must determine the number of arguments from the calling sequence and must transmit the argument address to the subprogram. An example is given below.

Calling

```
        BRM     SPROG
        PZE     5
        SNGL    arg1
        LOGL    arg2
EXTRA   DOUB    arg3
        DOUB    arg4
        DOUB    arg5
```

Receiving:

```
SPROG   PZE     0
        BRM     9SETUPV
        VARG    2
        ANY     0
        LOGL    0
        DOUB    *,X1
        STA     LOCN
        .
        .
        .
LOCN    PZE     0
```

The VARG 2 defines two fixed arguments

VARG specifies the number of fixed arguments (even if n = 0), and ANY and LOGL are the type specifications for the fixed arguments. The following line (DOUB*,X1) is the type specification for the variable arguments; it is always indirect and is normally indexed, to facilitate access of the remaining arguments. The 9SETUPV subprogram will process the fixed argument specifications as usual, but will replace the operand of the variable type specification with the address of the last fixed specification in the calling sequence (i.e., EXTRA - 1). The number of variable arguments is placed in LOCN†. Thus, for the above example, after 9SETUPV has been executed the receiving sequence looks like

```
SPROG   PZE     0
        BRM     9SETUPV
        VARG    2
        SNGL    arg1
        LOGL    arg2
        DOUB    *EXTRA-1,X1
        STA     LOCN
        .
        .
        .
LOCN    PZE     3
```

where arg_i means the effective address of argument i.

†The number of variable arguments may be zero.

Standard Calling/Receiving Sequences for No Arguments. The standard calling sequence for no arguments is

```
        BRM     subprogram
        PZE     0
```

The standard receiving sequence for no arguments is

```
subprogram  PZE     0
            BRM     9SETUP0
            PZE     0
```

Nonstandard Calling/Receiving Sequences. The compiler generates a nonstandard sequence whenever only one argument is required and the type of argument required by the subprogram is known. An example of a nonstandard calling sequence is

```
LDP     ARGUMENT
BRM     9SIN
```

A receiving sequence is not normally required; references such as 9SIN usually access the actual start of the subprogram.

The argument supplied is always located in the "principal register." Each type of data has its own principal register:

Type	Principal Register
Integer	A
Real	A, B
Double-precision	8DBL (or 8DBL0, 8DBL1, or 8DBL2)
Complex	8CPX (or 8CPXR, 8CPXI, 8CPX0, 8CPX1, 8CPX2, or 8CPX3)
Logical	A
Other (labels, etc.)	A

Receiving Sequence for Reentrant Subprograms.

The standard receiving sequence for reentrant subprograms is

```
$ENTRY  PZE
        DIR     *,1
        BRM     R\SETUP
        PZE     TEMPS
        .
        .
        .
EXIT    BRM     R\SETDWN
        BRR     ENTRY
FIRST   .       local variables, and temps
        .
        .
LAST    PZE     0
TEMPS   PZE     TEMPS-FIRST+1
```

Receiving Sequence for PROTECTED Subprograms

The standard receiving sequence for PROTECTED subprograms is

```
$ENTRY  PZE
        DIR     *,1
        BRM     R\SETUP
        PZE     *TEMPS
        .       normal receiving sequence and
        .       program
        .
EXIT    BRM     R\ENABLE
        BRR     ENTRY
FIRST   .       local variables, and temps
        .
        .
LAST    PZE     0
TEMPS   PZE     TEMPS-FIRST+1
```

The CONNECT Statement

The FORTRAN IV CONNECT statement generates code as described in Section 5.

MONITOR LINKAGE CELL R\MACH

R\MACH indicates the type of CPU that MONITOR is being used with. Bit 0 is set to a 0 if the CPU is either a 910 or a 925, otherwise it is set to a 1. Bit 1 is set to a 0 if the CPU is either a 910 or a 920, otherwise it is set to a 1. This code is illustrated in the following table:

CPU	Bit 0	Bit 1
910	0	0
920	1	0
925	0	1
930	1	1
9300	1	1

Note that the codes for the 930 and 9300 Computers are identical.

9. SYSTEM UPDATE ROUTINE

GENERAL DESCRIPTION

The UPDATE processor's purpose is to update the basic system tape for use by the system-generator (SYSGEN). It also may be used to update any file of a similar format. There are three basic functions of file maintenance, namely:

1. Replacement.
2. Insertion.
3. Deletion.

Other functions that the UPDATE processor can perform are:

1. File copying.
2. Labeling an output file
3. Rewinding a designated file or files.
4. Writing an end-of-file on a designated file or files.
5. Scanning a designated file for a given label.
6. Skipping a designated file forward or reverse.

UPDATE PROCESSOR CONTROL MESSAGES

The UPDATE processor is a control-message oriented routine. UPDATE is initiated by one of the following control messages:

△UPDATE from, to [= BLOCKED / UNBLOCKED]

△UPDATE to [= BLOCKED / UNBLOCKED]

where

from	=	a previously assigned symbolic file name, to be used as the input file.
to	=	a previously assigned symbolic file, to be used as the UPDATEd output file.
BLOCKED	=	Output file will be blocked ($\leq 400_{10}$ words/block).
UNBLOCKED	=	Output file will be unblocked (1 card image/block).

If neither BLOCKED nor UNBLOCKED are specified, BLOCKED is assumed.

The UPDATE control message which defines "from" and "to" implies a file copy from "from" to "to" with possible updating from the "C" device. If the UPDATE control message defines only one parameter (i.e., "to"), the UPDATE procedure is to copy from the "C" file to the "to" file.

If the UPDATE processor's first control message is

△END

the result is a file copy. This sequence implies that the UPDATE control message defined both the "from" and "to" tapes.

When the END control message is encountered, UPDATE terminates by copying the remainder of the "from" file onto the "to" file, provided that both the "from" and "to" fields on the UPDATE control message were defined. If the procedure is to copy from "C" to "to", UPDATE just terminates.

When the UPDATE processor's first control message is not END, the procedure followed depends on the particular UPDATE control message encountered.

UPDATE CONTROL MESSAGES

△REPLACE name

The REPLACE control message directs UPDATE to copy the "from" file up to the routine defined by the label "name". UPDATE then reads the next input image, to determine if it is a LABEL control message (see LABEL control message described in Section 2). If the control message is LABEL, the new label is written on the "to" file and UPDATE skips the label "name" and its entire routine on the "from" file. If the input image is not a LABEL, the label "name" from the "from" file is copied onto the "to" file and the remainder of the routine on the "from" file is skipped. The routine that is to be inserted (the replacement) is then copied from the "C" file to the "to" file.

△INSERT name

The INSERT control message direct UPDATE to copy the "from" file onto the "to" file, up through the label "name" and its corresponding routine. UPDATE then performs a "C"-to-"to" copy.

△DELETE name

The DELETE control message directs UPDATE to copy the "from" file onto the "to" file, up to "name". The label "name" and its entire routine are then skipped.

△SCAN name

The SCAN control message directs UPDATE to scan the "from" file for the label "name".

△SKIP ±nnnnn

The SKIP control message directs UPDATE to skip either forward[†] (+nnnnn) or reverse (-nnnnn) "nnnnn" records, where a record implies from one label record to another.

When UPDATE has completed its file maintenance function, it will provide a map of all of the labels that the "to" file contains.

[†]The "+" is not required when nnnn references a skip forward.

10. SYSTEM GENERATION

GENERAL DESCRIPTION

The system generation routine (SYSGEN) is a free-standing processor that will generate a real-time monitor system (MONITOR) either on a RAD File or on magnetic tape 1, channel A (or W, for the 900 Series). When SYSGEN is completed, the MONITOR system will operate from MT0A (or from MT0W, for the 900 Series) or the appropriate RAD unit.

SYSGEN is an absolute program with its own bootstrap and loader and is the first record on the SYSGEN tape, which is placed on tape 0, channel A (or W)[†]. The program is loaded into high core when a magnetic tape fill operation is performed.

The minimal peripheral requirements for SYSGEN are:

1. A magnetic tape on Channel A (MT0A or MT0W)
2. A typewriter on Channel A (TY1A or TY1W)
3. One of the following (system device):
   a. A RAD File
   b. A magnetic tape on channel A (MT1A or MT1W)

Optional peripherals are:

1. A card reader on channel A (CR1A or CR1W)
2. A line printer on channel A (LP1A or LP1W)

SYSGEN CONTROL MESSAGES

Before SYSGEN begins, the operator must supply the SYSGEN operating parameters by responding to several information request messages and by defining the system configuration of the ultimate MONITOR. These SYSGEN-time control messages provide the user with a dynamic system generation capability.

INPUT/OUTPUT REQUESTS

SYSGEN initially requests the device from which the control messages will originate, by typing the message:

INPUT FROM

The reply may be one of the following (followed by a period or carriage return):

1. TY (implies typewriter 1, channel A or W)
2. CR (implies card reader 1, channel A or W)

Any reply other than the above implies typewriter, and the error message

INPUT NOT 'TY' OR 'CR', 'TY' ASSUMED

will be typed.

SYSGEN then requests the device on which the control messages, diagnostics and other information will be displayed by typing:

OUTPUT ON

The reply may be one of the following (followed by a period or carriage return):

1. TY (implies typewriter 1, channel A or W)
2. LP (implies line printer 1, channel A or W)

Any reply other than the above implies typewriter, and the error message

OUTPUT NOT 'TY' OR 'LP', 'TY' ASSUMED

will be typed.

MONITOR SYSTEM CONFIGURATION

The first control message required by SYSGEN must supply information about the MONITOR system configuration. The form of this message is given below[†].

ΔBASE-MACHINE, SYSTEM-DEVICE, DISC-SIZE, CHECK

BASE-MACHINE is either 910, 920, 925, 930, or 9300

SYSTEM-DEVICE is MT0A(MT0W) or DFnc (where: 1 n 2 and c=A, W, B, Y, C, ..., H)

DISC-SIZE is the number of characters on a RAD File, nnCK (needed only if the SYSTEM-DEVICE is a RAD File);
where: nn = 5, 10, 26, 68, etc.,
C = nn* 100, and
K = C * 1000
e.g., 5CK (i.e., 500,000)

CHECK is an optional request for SYSGEN to read the system output just written and to compare it with that which should have been written.

Typical messages are

Δ910, MT0W, CHECK.
Δ930, MT0A.
Δ9300, DF1A, 47CK, CHECK.

[†]All references herein to any device on channel A or B also apply to channel W or Y for the 900 Series Computers.

[†]All SYSGEN control messages begin with a Δ in character position 1 and end with either a period, a carriage return, or a maximum length of 80 characters.

RESIDENT I/O DRIVER REQUESTS

The second control message required by SYSGEN supplies the information concerning what peripheral device drivers are to be resident at all times for this particular MONITOR system.

The control message appears as one of the following:

ΔDRIVERS DEVICE-CODE(S)
ΔDRIVERS* DEVICE-CODE(S)

where:

DEVICE-CODE(S)[t] = blank (the typewriter, NO I/O operation, and the SYSTEM-DEVICE drivers are automatically assumed, even if the field is not blank but does not request these three drivers).

= DF and/or MD, implies disc/drum driver.

= TY and/or PR and/or PP, implies typewriter/paper-tape driver.

= CR and/or CP, implies card reader/card punch driver.

= LP, implies line printer driver.

= MT, implies magnetic tape driver.

= PL and/or NO, implies NO I/O operation.

DRIVERS — implies that I/O error recovery during real-time processing is requested.

DRIVERS* — implies that no I/O error recovery during real-time processing is requested.

Typical messages are

ΔDRIVERS MT, CR, CP, TY
ΔDRIVERS* DF, MT, NO
ΔDRIVERS

SYSTEM STANDARD ASSIGNMENT GENERATION

SYSGEN proceeds to load the resident MONITOR and INSTALlation package according to the BASE-MACHINE designation. The INSTALlation package contains the information needed for dynamic generation of the Unit Availability Table (UAT), Unit Name Table (UNT), standard assignment portion of the resident symbol table (SYMTAB), and the FCBs necessary to manage the system files.

[t]Each field must be separated by a comma.

The UNT contains the device names for all devices defined by INSTALlation and a corresponding UAT entry which contains the channel and device number bit settings. Each entry in the UAT also references its particular I/O driver.

STANDARD ASSIGNMENT MODIFICATIONS

After generation of SYMTAB, additional standard assignments can be added to SYMTAB as well as modifications to all the existing standard assignments, except for

R\PERM (permanent file)
R\PROC (processor file)
R\PROK (META-SYMBOL Proc deck file)
R\PRIL (primary library file)
R\SECL (secondary library file)
R\SWAP (swap file)
R\CONS (system console file)
NO (NO I/O operation)

by modification control messages of the form

ΔASSIGN STANDARD-ASSIGN-NAME = ASSIGNMENT

where:

STANDARD-ASSIGN-NAME = from 1- to 8-character operational label (e.g., ABC, MXYZT).

ASSIGNMENT = a 4-character device name which is defined in UNT, or device NO.

= a 1- to 8-character name which may be defined in UNT or, if not in UNT, must be in SYMTAB as a previously defined STANDARD-ASSIGN-NAME

Typical examples are

ΔASSIGN MX = LP1A
ΔASSIGN 10 = MX

As many of these modification or additional standard assign control messages may be supplied as are needed.

The modification control messages are terminated by a FIN control message of the form

ΔFIN

USER HOLD FILES (DISC)

If the SYSTEM-DEVICE is a RAD File, SYSGEN needs to know if the RAD File contains user-defined "HOLD" files. SYSGEN will type:

ARE THERE HOLD FILES

The answer to be typed in is either YES or NO. If the answer is YES, the HOLD files are retained when generating the MONITOR system. If the answer is NO, the MONITOR system is generated and the remainder of the RAD File is not preserved. If the response is neither YES nor NO (e.g., YAS, N0), SYSGEN will continue with the request until the response is YES or NO.

GENERAL SYSTEM GENERATION

SYSGEN determines which I/O drivers are required (and what the BASE-MACHINE is) and will load them as part of the basic resident MONITOR.

SYSGEN analysis is then completed, except for generating the MONITOR system on the SYSTEM-DEVICE. The BASE-MACHINE bootstrap is loaded and written on the SYSTEM-DEVICE. The MONITOR is then written, followed by the INSTALlation information.

SYSGEN can then use the MONITOR for all of its I/O for the remainder of system generation. The remainder of the system generation depends on the BASE-MACHINE to determine whether the remainder of the MONITOR system is for a 900 Series or a 9300 Computer.

The appropriate overlay loader is loaded into core to serve as the loader for the remainder of the system generation. The MONITOR is then informed that SYSGEN is to be the executive system. All of SYSGEN is released from core for loading purposes, except for the executive control.

The SYSGEN input tape (MT0A) is positioned to the I/O drivers. All MONITOR routines, processors, and I/O drivers which were not previously loaded as part of the basic MONITOR are loaded and put on the processor file (R\PROC) in a semi-absolute format. When the META-SYMBOL Proc decks are encountered, they are put on the Proc deck file (R\PROK).

SYSGEN continues by generating the primary library (R\PRIL) and secondary library (R\SECL) files in semi-absolute format.

When the ENDGEN record is read from the input tape (MT0A), SYSGEN completes the SYSTEM-DEVICE initialization. MONITOR is then informed that SYSGEN has finished and that the MONITOR executive routine is in control.

SUMMARY OF SYSGEN MESSAGES

SYSGEN contains various error, I/O, and general-information messages which may occur during a SYSGEN operation.

INPUT/OUTPUT

The following table defines the type of message, its input source, and where it is displayed (see INPUT FROM and OUTPUT ON messages, discussed in "Input/Output Requests."):

Input From	Out. On	Message Type	Where Displayed
TY	TY	Input control message	Not displayed
		Error message	On TY1A
		Installation map	On TY1A
TY	LP	Input control message	On LP1A
		Error message	On TY1A and LP1A
		Installation map	On LP1A
CR	TY	Input control message	On TY1A
		Error message	On TY1A
		Installation map	On TY1A
CR	LP	Input control message	On LP1A
		Error message	On TY1A and LP1A
		Installation map	On LP1A

ERROR AND I/O MESSAGES

The SYSGEN error and I/O messages are as follows:

Message	Cause	Results/Correction
NO 'CONTROL INFO'	The first control message (BASE-MACHINE, SYSTEM-DEVICE, etc.) is completely blank.	SYSGEN will halt with A-reg.=01 Determine correct control message and clear halt
'CONTROL INFO' NOT COMPLETE	The first control message (see above) does not contain a SYSTEM-DEVICE	SYSGEN will halt with A-reg. =02. Determine correct control message and clear halt.
DISC SIZE UNKNOWN, OR NOT DEFINED	The first control message (see above) defines SYSTEM-DEVICE as DFnc(disc) and the size is either unknown or not present. (where: $1 \leq n \leq 2$ and c = A, W, B, ..., H)	SYSGEN will halt with A-reg. =014. Determine correct control message and clear halt.

Message	Cause	Results/Correction
OUTPUT DEVICE NOT 'DISC' OR 'MT0A-W'	The first control message (see above) has an illegal SYSTEM-DEVICE definition. Must be MT0A, MT0W, or DFnc where: 1≤n≤2 and c=A, W, B, Y, ..., H.	SYSGEN will halt with A-reg. =03. Determine correct control message and clear halt.
BASE-MACHINE I.D.UNKNOWN	The first control message BASE-MACHINE is not defined as 910, 920, 925, 930, or 9300.	SYSTEN will halt with A-reg. =013. Determine correct control message and clear halt.
ASSIGN/FIN 'CONTROL INFO' MISSING	A modification assign control message is incomplete, or the control message was supposed to be "ΔFIN".	SYSGEN will halt with A-reg. =04. Determine correction and clear halt.
NO DRIVER CONTROL MESSAGE	The second control message required by SYSGEN must be the control message DRIVERS.	SYSGEN will halt with A-reg. =016. Determine what the control message should be (even if it is just 'Δ DRIVERS') and clear halt.
XX IS ILLEGAL DRIVER REQUEST	The DRIVERS control message has requested an I/O driver("XX") which is unknown to SYSGEN.	SYSGEN will ignore the request and will continue processing the remainder of the control message.
POSSIBLE SYS-GEN ERROR (5)	SYSGEN internal information missing when generating UAT. Caused by a possible hardware malfunction.	SYSGEN will continue as though no error had occurred. If the error is determined to be catastrophic, the entire SYSGEN operation should be restarted. If the error persists, check the hardware.
XXXX DEVICE NOT AVAILABLE	When generating the symbol table for standard assignments, a standard assignment references a nonexistent device "XXXX".	Standard assign from INSTALlation is ignored and the generation continues.
MOD.OF A NON-STANDARD ASSIGN SYMB.	A modification control message requests an assignment modification to a symbol which is not a standard assign.	SYSGEN ignores the modification and continues processing by obtaining the next control message.
MOD.OF A NON-DF/MD ASSIGN, TO=DF/MD	A modification control message requests a modification of a standard assignment (which was not originally assigned to a disc or drum) to be assigned to a disc or drum.	SYSGEN ignores the modification and continues processing by obtaining the next control message
MOD.OF STAND.ASSIGN, DEV.NOT FOUND	A modification control message requests a modification of a standard assign and the peripheral device is unavailable.	SYSGEN ignores the modification and continues processing by obtaining the next control message.
ILLEGAL MOD.TO STAND.ASSIGN/SYMB.	A modification control message requests an assignment which refers to a standard assignment which cannot be assigned to.	SYSGEN ignores the modification and continues processing by obtaining the next control message.
INPUT NOT 'TY' OR 'CR', 'TY' ASSUMED	Incorrect response to the initial SYSGEN request: INPUT FROM.	SYSGEN assumes 'TY1A' as its input and continues.
OUTPUT NOT 'TY' OR 'LP', 'TY' ASSUMED	Incorrect response to the initial SYSGEN request: OUTPUT ON.	SYSGEN assumes 'TY1A' as its output and continues.
MTnA ERROR	(0 ≤ n ≤ 1) magnetic tape read or write error.	SYSGEN halts with A-reg.='MTE' (446325). Clear halt for retry. If error persists, check tape and tape drive.

Message	Cause	Results/Correction
MTnA NOT READY	(0 ≤ n ≤ 1) magnetic tape is either physically not ready or is file-protected for a write attempt.	SYSGEN waits for the condition to be corrected and then continues.
CR1A ERROR	Card reader error; e.g., validity check, feed check, or read check.	SYSGEN halts with A-reg.='CRE' (235125). Correct condition and clear halt to continue.
CR1A NOT READY	Card reader is not ready.	SYSGEN waits for the condition to be corrected and then continues.
LP1A ERROR	Line printer error.	SYSGEN halts with A-reg.='LPE' (434725). Correct condition and clear halt to continue.
LP1A NOT READY	Line printer is not ready or on line.	SYSGEN waits for the condition to be corrected and then continues.
DFnc ERROR	Disc n (1 ≤ n ≤ 2) on channel c (=A, W, B, Y, C, ..., H) read or write error.	SYSGEN halts with A-reg.='DFE' (242625). Clear halt for retry. If error is persistent, check disc for hardware problems.
DFnc NOT READY	Disc n(1 ≤ n ≤ 2) on channel c (=A, W, B, Y, C, ..., H) not ready.	SYSGEN waits for the condition to be corrected and then continues.
PERIPHERAL-DEVICE XXXX NOT AVAILABLE	A peripheral device which has been defined for SYSGEN I/O use is not available (XXXX = device defined).	SYSGEN cannot continue. The entire SYSGEN operation must begin over, with the correct devices defined.

GENERAL INFORMATION MESSAGES

The general information messages produced by SYSGEN constitute a map of INSTALlation. This map displays the information contained in the UNT, UAT, and SYMTAB tables and also the MONITOR system characteristics.

Monitor System Characteristics

The following is an example of the MONITOR system characteristics:

Case 1: (where SYSTEM-DEVICE = DF1A)

```
BASE MACHINE = 930
SYSTEM DEVICE= DF1A
** ECHO-CHECK
R\MACH       = 60000000
SECTOR MAP   = 003774 (DISC-ADDRESS)
TOTAL SECTORS= 004000=(*100 WORLS PER SECTOR *4 CHAR.PER WORD=NO.CHAR.(OCTAL))
```

Case 2: (where SYSTEM-DEVICE = MT0A)

```
BASE MACHINE = 925
SYSTEM DEVICE= MT0A
NO ECHO-CHECK
R\MACH       = 20000000
```

BASE-MACHINE = The machine in which the MONITOR system is to operate.

SYSTEM-DEVICE = The peripheral device where the MONITOR system resides.

**ECHO-CHECK = SYSGEN checks what was written on the SYSTEM-DEVICE against that which should have been written.

NO ECHO-CHECK = As above, except that no check is made.

R\MACH = A flag word used by the MONITOR I/O drivers and system processors to determine if I/O error recovery during real-time processing is requested, and also what the BASE-MACHINE is.

SECTOR MAP = Address (octal) of the disc sector where the MONITOR disc-sector-map is maintained.

TOTAL SECTORS = The total number (octal) of sectors that this particular disc contains.

Unit Name Table Map (UNT)

The following is an example of the UNT map:

U N T

```
ADDRS: DEVICE :ADDRS:CHAN-DEV:I/O: DRIVER
34250   DF1W   34026 00000026 00   R\DISC
34252   MT2A   34030 00000027 00   R\DISC
34254   MT2W   34030 00000027 00   R\DISC
34262   CR1A   34034 00000046 06   R\CARD
34264   CR1W   34034 00000046 06   R\CARD
34316   MT1A   34052 00000011 14   R\TAPE
34320   MT1W   34052 00000011 14   R\TAPE
34322   MT2A   34054 00000012 14   R\TAPE
34324   MT2W   34054 00000012 14   R\TAPE
34326   MT3A   34056 00000013 14   R\TAPE
34330   MT3W   34056 00000013 14   R\TAPE
34332   MT4A   34060 00000014 14   R\TAPE
34334   MT4W   34060 00000014 14   R\TAPE
34336   MT5A   34062 00000015 14   R\TAPE
34340   MT5W   34062 00000015 14   R\TAPE
34342   MT6A   34064 00000016 14   R\TAPE
34344   MT6W   34064 00000016 14   R\TAPE
34346   MT7A   34066 00000017 14   R\TAPE
34350   MT7W   34066 00000017 14   R\TAPE
34352   LP2E   34070 00000127 00   R\DISC
34354   LP2Y   34070 00000127 00   R\DISC
34362   PP2E   34074 00000105 03   R\PAPR
34364   PP2Y   34074 00000105 03   R\PAPR
34440   MTCE   34150 04000010 14   R\TAPE
```

where:

ADDRS (1st)	=	Core address of UNT's DEVICE entry
DEVICE	=	Actual name of DEVICE
ADDRS (2nd)	=	DEVICE reference address to UAT
CHAN-DEV	=	The channel EOM bits with UNIT NUMBER
I/O	=	Index to the I/O DRIVER name table
DRIVER	=	The DEVICE name is connected to its appropriate I/O DRIVER

Symbol Table Map (SYMTAB)

The following is an example of the SYMTAB map:

S Y M T A B

```
CHAIN:CODE:<LG>:<FCB=2>:  NAME  :<FCB=6>:ADDRS: DEVICE :ADDRS:CHAN-DEV:I/O: DRIVER

35031   02    NO     YES    00000001    NO    34354   MT7W   34072 00000017 14   R\TAPE
35025   02    NO     YES    12345678    NO    34354   MT7W   34072 00000017 14   R\TAPE
35021   02    NO     YES    00010000    NO    34354   MT7W   34072 00000017 14   R\TAPE
35015   02    NO     YES    5           NO    34354   MT7W   34072 00000017 14   R\TAPE
34771   02    NO     YES    VWXYZ       NO    34262   CR1A   34036 00000006 06   R\CARD
34765   02    NO     YES    ABCDEFGH    NO    34262   CR1A   34036 00000006 06   R\CARD
34745   02    NO     YES    H           NO    34250   NO     34030 00000000 22   R\NO
34651   13    NO     NO     R\SWAP      YES   35117   DF1A   34026 00000026 00   R\DISC
34617   02    NO     YES    E0          NO    34250   NO     34030 00000000 22   R\NO
34613   02    NO     YES    E1          NO    34262   CR1A   34036 00000006 06   R\CARD
34564   12    NO     NO     X1          YES   35115   DF1A   34032 00000026 00   R\DISC
34554   02    NO     YES    R\OVRL      NO    34320   MTOW   34054 00000010 14   R\TAPE
34550   03    NO     YES    R\SECL      NO    34320   MTOW   34054 00000010 14   R\TAPE
34544   03    NO     YES    R\PRIL      NO    34320   MTOW   34054 00000010 14   R\TAPE
34540   03    NO     YES    R\PROR      NO    34320   MTOW   34054 00000010 14   R\TAPE
34024   03    NO     YES    R\PROC      NO    34320   MTOW   34054 00000010 14   R\TAPE
33547   10    NO     NO     R\IOCP      NO    33441
33543   10    NO     NO     R\IOTB      NO    33451
33537   10    NO     NO     R\OPRL      NO    33511
33533   10    NO     NO     R\MACP      NO    33510
33527   10    NO     NO     R\CERR      NO    33506
33523   10    NO     NO     R\LUNT      NO    33774
33517   10    NO     NO     R\UNT       NO    33773
33513   10    NO     NO     R\UAT       NO    33777
00006   10    NO     NO     R\REST      NO    33512
```

where:

CHAIN	=	the core address of the next symbolic NAME in the SYMTAB map.
CODE	=	the code given for the various types of SYMTAB entries, e.g., 02 = FCB reference (not disc or drum and is reassignable) 12 = FCB reference originally to a disc or drum (is reassignable)

03 = SYSTEM-FCB reference (cannot be reassigned nor can a file name be assigned to it)
13 = SYSTEM-FCTB reference (same as for code 03, but this reference is ignored by SYSGEN if the SYSTEM-DEVICE is MT0A.
10 = MONITOR and INSTALlation entries.

<LO> = NO if "NAME" is not same file as "LO" file.
= YES if "NAME" is the same file as "LO".

<FCB=2> = NO if "NAME" is not an FCB reference.
= YES if "NAME" is a 2-word FCB reference to the UAT.

NAME = The 1-8 character symbolic name.

<FCB=6> = NO if "NAME" is not an FCB reference.
= YES if "NAME" is a 6-word FCB reference (disc or drum).

ADDRS(1st) = The core address of UNT's DEVICE entry if "NAME" is an FCB reference.
= The core address of the MONITOR or INSTALlation definition if "NAME" is not an FCB reference.

Note: The remainder of a SYMTAB entry information line is blank if the "NAME" is not an FCB reference. (See UNT example for explanation of remainder of a SYMTAB entry's information line if "NAME" is an FCB reference.)

SYSGEN EXAMPLES

The following are examples of a SYSGEN operation with modifications to standard assignments and subsequent addition of standard assigns:

Example 1:

```
    INPUT FROM     TY.
    OUTPUT ON      LP.
△910   DF1A,5CK.
△DRIVERS.                    (DF, TY, and NO assumed)
△ASSIGN  A = DF1A
△ASSIGN TEMPORARY = A  (actual file name = TEMPORAR)
△ASSIGN  X3 = NO
△FIN
```

Example 2:

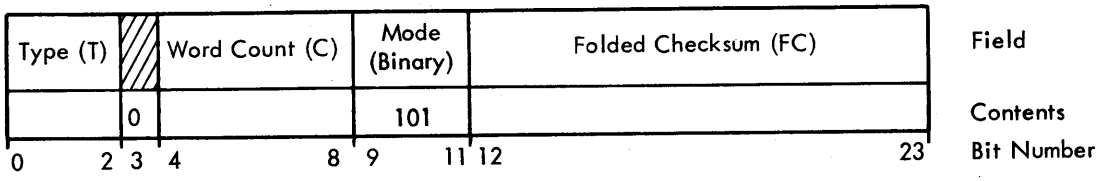
```
    INPUT FROM     CR.
    OUTPUT ON      TY.
△9300  MT0A,CHECK.
△DRIVERS   DF,CR,NO.      (MT, and TY assumed)
△ASSIGN  X1 = MT1A
△ASSIGN  X2 = MT2A.
△ASSIGN  X3 = MT2A.
△ASSIGN 12345678 = X3
△ASSIGN  X4 = 12345678
△FIN
```

APPENDIX A
SDS STANDARD BINARY LANGUAGE

The following description specifies a standard binary language for SDS 900 Series and 9300 Computers. This language is intended to be both computer-independent and medium-independent. Thus, the language provides for handling Programmed Operator definitions and references, even though the 9300 Computer does not have this hardware feature; similarly, there is a provision for relocation relative to blank COMMON.

In the following description of the language, a file is the total binary output from the assembly/compilation of one program or subprogram. A file is both a physical and logical entity, since it can be subdivided physically into unit records and logically into information blocks. While a unit record (in the case of cards) may contain more than one record, a logical record may not overflow from one unit record to another.

1. CONTROL WORD – first word in each type of record



T	Record Type
000	Data record (text)
001	External references and definitions, block and program lengths
010	Programmed Operator references and definitions
011	End record (program or subroutine end)
101	Data Statement Record
	(other codes unassigned)

C = total number of words in record, including Control Word

Note that the first word contains sufficient information for handling these records by routines other than the loader (that is, tape or card duplicate routines). The format is also medium-independent, but preserves the Mode indicator positions desirable for off-line card handling.

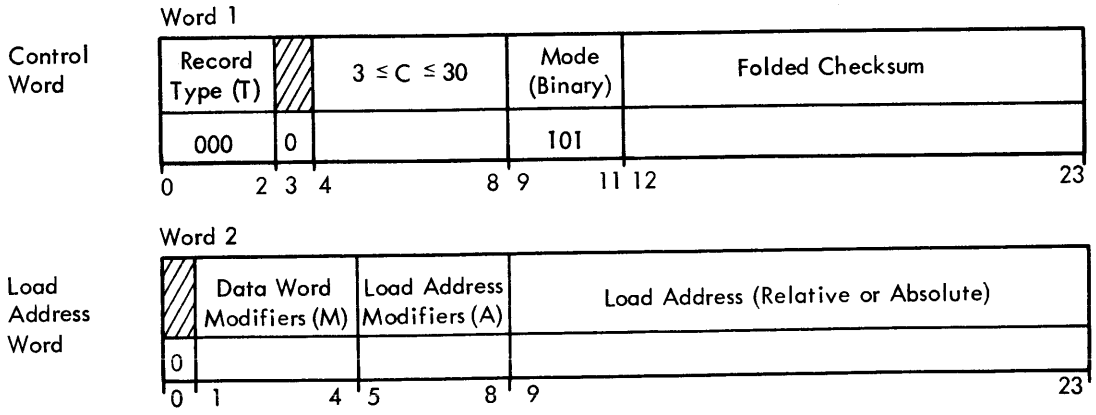
An exclusive OR checksum is used. If the symbol ⓤ is used to denote exclusive OR, and W_i denotes the ith word in the record ($1 \leq i \leq C$), then

$$FC = (W_1)_{0-11} \text{ⓤ } (S)_{0-11} \text{ⓤ } (S)_{12-23} \text{ⓤ } 07777$$

where

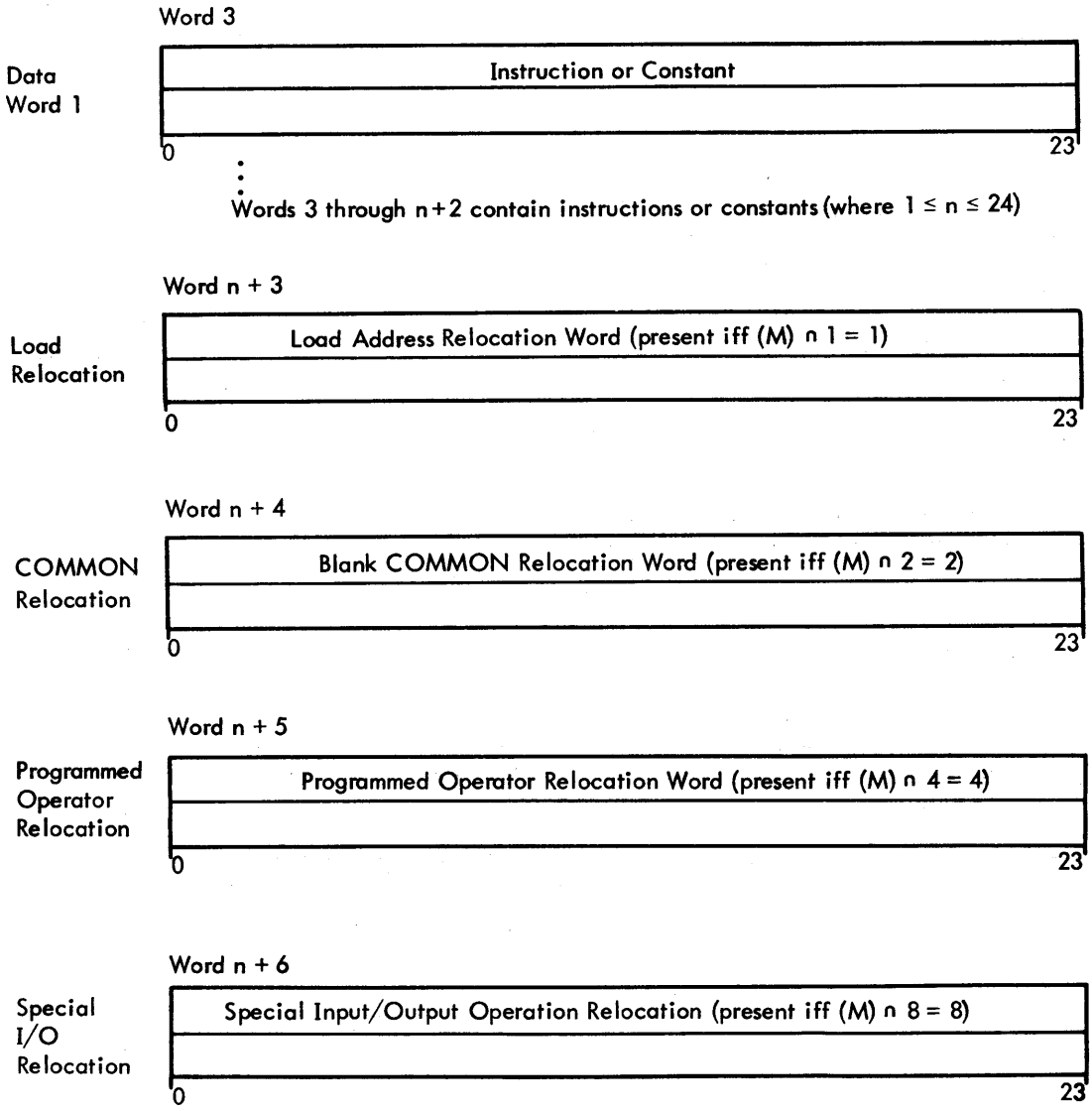
$$S = W_2 \text{ⓤ } W_3 \text{ⓤ } \cdots \text{ⓤ } W_c$$

2. DATA RECORD FORMAT (T = 0)



The presence of bits in field M indicates the presence of words n+3, n+4, n+5, and n+6 (shown below):

If bit position 4 contains a 1, word n+3 (load relocation) is present.
If bit position 3 contains a 1, word n+4 (COMMON relocation) is present.
If bit position 2 contains a 1, word n+5 (POP relocation) is present.
If bit position 1 contains a 1, word n+6 (special I/O relocation) is present.



Words n+3 through n+6 are modifier words. Each bit in each of these words corresponds to a data word; that is, bits 0 through 23 of each modifier word correspond to data words 3 through n+2 (where 1 ≤ n ≤ 24.) A bit set to a 1 in a modifier word indicates that the specified data word requires modification by the loader. There are four types of modification (and, hence, four possible modifier words) which are indicated in data records. The presence of a modifier word in a data record is indicated by the M (data word modifier) field in the load address word.

The load address is subject to modification, as indicated by the A field of the load address word, as follows:

(A) 0, absolute

(A) ∩ 1 = 1, current load relocation bias is added to load address

(A) ∩ 2 = 1, current COMMON relocation bias is added to load address; the remaining bits of A are unassigned

(A) = 3, illegal

3. EXTERNAL REFERENCES AND DEFINITIONS, BLOCK AND PROGRAM LENGTHS (T = 1)
(Includes labeled COMMON, blank COMMON and program lengths)

Control Word

Word 1

Record Type (T)		4 ≤ C ≤ 31*	Mode (Binary)	Folded Checksum
001	0		101	
0 2	3	4 8	9 11	12 23

* From 1 to 10 items per record

COMMON Length or Program Length Item

1- to 8-Character Label

C1	C2	C3	C4
C5	C6	C7	C8
0 5	6 11	12 17	18 23

Length Word

Item Type		B	C	Length of Program or COMMON Block (L)
00	00000			
0 1	2 6	7	8	9 23

B = 1 if (L) is program length
C = 1 if (L) is length of a labeled COMMON block

External Reference Item

1- to 8-Character Label

C1	C2	C3	C4
C5	C6	C7	C8
0 5	6 11	12 17	18 23

Chain Word

Item Type		Address Modifiers (A)**	Address of Last Reference
01	000		
0 1	2 4	5 8	9 23

** See data record, load address word, for interpretation.

External Definition Item

1- to 8-Character Label

C1	C2	C3	C4
C5	C6	C7	C8
0 5	6 11	12 17	18 23

Value Word

Item Type		Address Modifiers (A)**	Absolute or Relocatable Value
10	000		
0 1	2 4	5 8	9 23

** See data record, load address word, for interpretation

External symbolic definitions include subroutine "identification" as a subset and require no special treatment of subroutines with multiple names.

External Reference with Addend Word*

1- to 8-Character Label

C1	C2	C3	C4
C5	C6	C7	C8

0 5 6 11 12 17 18 23

Chain Word

Item Type		Address Modifiers (A)**	Address of Last Reference
11	000		

0 1 2 4 5 8 9 23

Addend Word

	Value of Addend
000 000 000	

0 8 9 23

* One of these items for each unique reference; e.g., each of the following references is represented by a separate item: A+5, B+5, B+6, C+2, C+5

** See data record, load address word, for interpretation.

4. PROGRAMMED OPERATOR REFERENCES AND DEFINITIONS (T = 2)

Control Word

Word 1

Record Type (T)		4 ≤ C ≤ 31**	Mode (Binary)	Folded Checksum
010	0		101	

0 2 3 4 8 9 11 12 23

**From 1 to 10 items per record

Internal Programmed Operator

1- to 8-Character Mnemonic

C1	C2	C3	C4
C5	C6	C7	C8

0 5 6 11 12 17 18 23

Item Type	Sequence No.	R	Origin of Programmed Operator Routine
00		1	

0 1 2 7 8 9 23

Programmed Operator Reference

1- to 8-Character Mnemonic

C1	C2	C3	C4
C5	C6	C7	C8

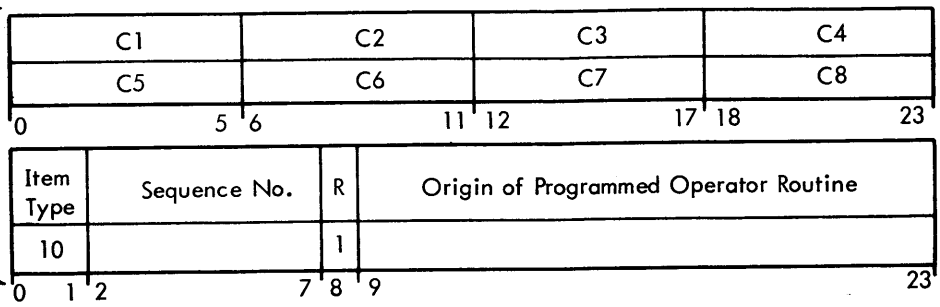
0 5 6 11 12 17 18 23

Item Type	Sequence No.	R	
01		0	000 000 000 000 000

0 1 2 7 8 9 23

External Programmed Operator Definition

1- to 8-Character Mnemonic

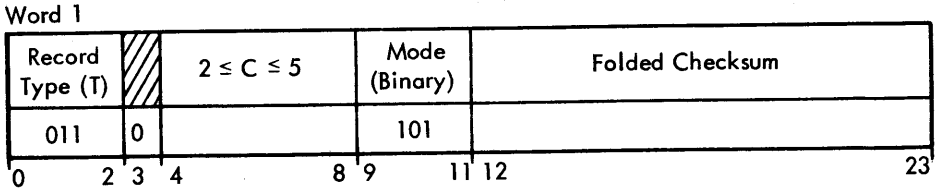


R = 1 iff origin of Programmed Operator routine is relocatable. The sequence number indicates the order in which the definitions or references occurred in the source program.

5. END RECORD (T = 3)

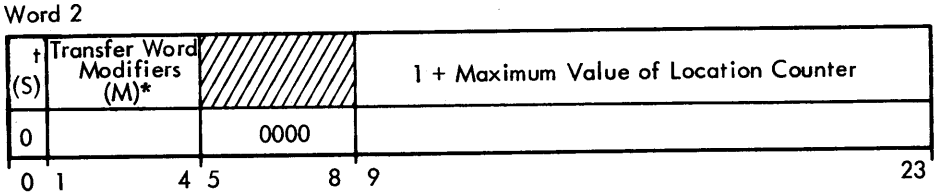
Control Word

Word 1



Length of Program

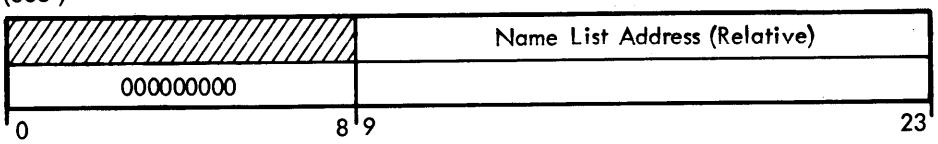
Word 2



* See data record description for interpretation.

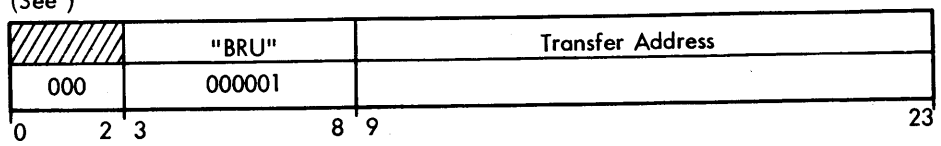
Name List Location Word

(See[t])



Transfer Word

(See[t])

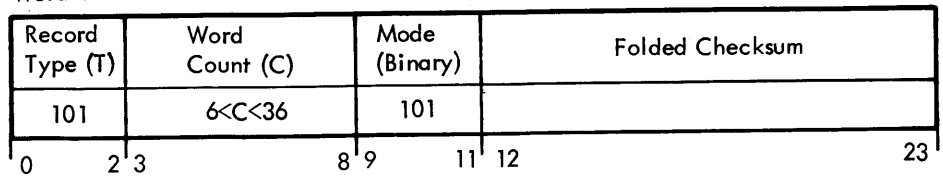


This may be followed by modifier words.

6. DATA STATEMENT RECORD FORMAT (T = 5)

Control Word

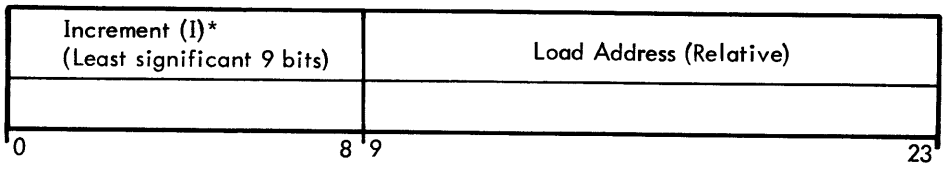
Word 1



[t] If S = 1, Word 3 is the Name List Location Word and Word 4 is the Transfer Word.
If S = 0, Word 3 is the Transfer Word, and the Name List Location Word is omitted.

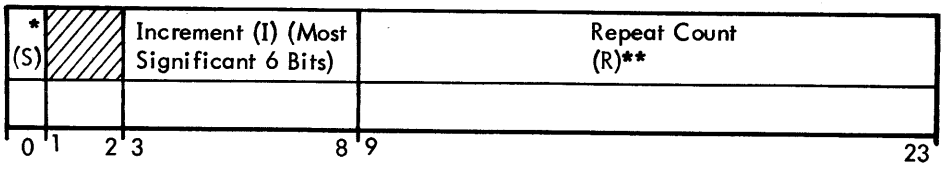
Word 2

Load Address Word



* The increment (I) is added to the relative load address to obtain the next relative load address for a repeat load.

Repeat Count Word



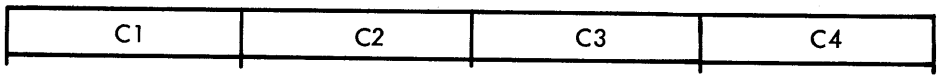
* If S = 1, words 6 through C ($6 \leq C \leq 36$) are loaded relative to the labeled COMMON block's origin.

If S = 0, words 4 through C ($4 \leq C \leq 36$) are loaded relative to the subprogram origin.

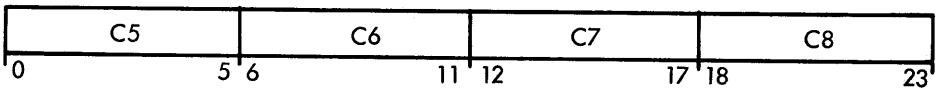
** Data words 4+2S through C are repeatedly loaded (R) times in increments of (I)

Word 3+S

COMMON Block Label

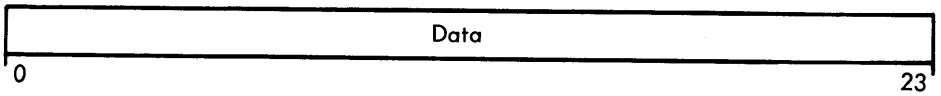


Word 3 + 2S



Word 4 + 2S

Data Word 1



Words 4+2S through C contain constants

7. BINARY CARD ORDERING

The loader places certain restrictions on the permissible ordering of relocatable binary cards. That is, the required ordering is as follows:

a. Type 1 cards (containing definitions of external symbols) and Type 2 cards must physically precede all other cards (except Type 5; see below).

b. Type 0 cards must follow Type 1 and Type 2 cards.

c. Type 1 cards containing references to externally defined symbols must then follow type 0 cards.

d. The last card in any deck must be a Type 3 card.

e. Type 5 cards may appear anywhere, prior to the Type 3 card, as long as they follow the definition of the item into which data is to be loaded.

APPENDIX B
900 SERIES REAL-TIME TAPE MONITOR

The tape version of MONITOR is functionally similar to the disc version, but operates in a magnetic tape environment without the mass storage and rapid access facilities of RAD Files. It provides interrupt and batch processing capabilities for real-time and general-purpose applications where RAD Files are not required.

The "swapping out" of batch jobs is optional and, if desired, a magnetic tape unit must be dedicated for this purpose. Dynamic loading of programs during interrupt processing is considerably slower than for the disc version, due to the differences in tape and disc access and transfer rates.

Library search and program load times are dependent on the number and size of the library programs.

A minimum of two magnetic tape units are required, and, if the batch "swapout" facility is to be used, a third magnetic tape unit is needed. Also, if process-and-GO is desired or if META-SYMBOL encoded decks are to be assembled with symbolic corrections, then an additional magnetic tape unit is necessary. (See the table below.)

The minimum hardware configuration for the tape MONITOR is identical to that for the RAD MONITOR, except that a magnetic tape unit is substituted for the RAD unit.

Function	Is Function Required By Job?											
Any assembly or compilation other than META-SYMBOL assembly of encoded program unit with symbolic corrections (see Section 1).	No	No	No	Yes	No	Yes	Yes		Yes			
META-SYMBOL assembly of encoded program unit with symbolic corrections (see Section 4).	No	No	No	No	No	No	No	Yes	No	Yes	Yes	Yes
GO output and/or LOAD input (see Section 3).	No	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	Yes
Swapping of batch jobs during real-time operations (see Section 5).	No	Yes	No	No	Yes	Yes	No	No	Yes	Yes	No	Yes
Number of tape units required[t]	1[tt]	2	2	2	3	3	3	3	4	4	4	5

[t]Includes only those tape units required by MONITOR.
[tt]Real-time operations only; no batch processing to be done.

INDEX

- A -

Abort routine (see System abort routine)
Accounting routines, 3
ASA compatibility, 2, 6
ASSIGN control message, 3, 23, 34

- B -

BACKSPACE control message, 8
BASE-MACHINE designation, 33, 35, 36, 37
Batch processing, 1, 10, 19, 23, 24, 47
BI file, 4, 6, 7, 11
Binary card ordering, 46
Binary output, 12
Blank COMMON, 41, 43
Blocked files, 32
Bootstrap program, 2, 35
Branch trace, 16
Branch and mark trace, 16

- C -

C device (see Control message input device)
Calling sequences (see Standard calling sequences)
Card reader/punch operations, 21
Carriage control, 18, 20, 22
CEASE control message, 17
Channel number, 18, 19
Channel-active test, 19
CHECK designation, 33, 39
Comment cards, 6
Comments field, 7
Compile-and-go operations, 1, 12, 47
Concordance listing, 5
CONNECT control message, 9, 15
CONNECT statement, 29, 31
Control message input device, 4, 8, 19, 32
Control message scan routine, 9, 24, 25
Control messages, 1, 3
  ASSIGN, 3, 23, 34
  BACKSPACE, 8
  CEASE, 17
  CONNECT, 9, 15
  DATA, 8
  DATE, 4
  DEBUG, 16
  DELETE, 32
  DISPLAY, 17
  DRIVERS, 34
  DUMP, 17
  END, 32
  ENDFILE, 8
  EOF, 8, 12, 13, 14
  FIN, 8, 34
  FORTRAN, 6
  INCLUDE, 7, 10, 16
  INSERT, 16, 17, 32
  JOB, 3
Control messages (cont.)
  LABEL, 5, 32
  LEAVE, 16
  LOAD, 4, 5, 6, 7, 8, 10, 16
  MESSAGE, 4
  METAXXXX, 5, 12
  PAUSE, 5, 8, 17, 24
  RELEASE, 3, 4
  RENTAB, 17
  REPLACE, 32
  REWIND, 8
  SCAN, 32
  SEG, 7, 10
  SKIP, 32
  SNAP, 16
  SYMBOL, 5, 6
  SYMTAB, 17
  TITLE, 4
  TRACE, 16
  TRAP, 16
  UPDATE, 32
Control word, 41
Core memory requirements, 2, 47
Counters, 2

- D -

DATA control message, 8
Data record, 41
Data statement record, 41, 45
Data type codes, 27
DATE control message, 4
DEBUG control message, 16
Debugging, 1, 9, 16
DELETE control message, 32
Delimiter table, 25
Device-independent files, 1
Diagnostic routines (see Debugging)
Disc files (see RAD Files)
Disc sector map, 23, 24
Disc sector map search routine, 23, 24
DISC-SIZE designation, 33
DISPLAY control message, 17
DRIVERS control message, 34
DUMP control message, 17
Dynamic loading, 1, 47
Dynamic storage, 2, 26

- E -

Empty-sector pool, 23, 24
Encoded input, 4, 5, 8, 47
END control message, 32
End record, 41, 45
End-action routine, 18, 19
ENDFILE control message, 8
ENDGEN record, 35
EOF control message, 8, 12, 13, 14
EOM instruction, 18, 19, 38
Error flags, 19, 21, 22, 23

Error messages, 2
Executive, 1, 2, 17, 35
Exit routine (see System exit routine)
External definitions, 7, 11, 16, 41, 43, 44, 46
External references, 15, 41, 43, 44, 46

- F -

File Control Blocks (FCBs), 4, 18, 19, 21, 23, 24
File Description Tables (FDTs), 18, 19, 20, 21, 22, 23
File maintenance, 2, 32
FIN control message, 8, 34
Fixed segments, 7
FORTRAN, 1, 2, 8, 12, 22, 28
FORTRAN control message, 6
FORTRAN IV I/O specifications, 6
Full trace, 16

- G -

GLOBAL variables, 9
GO file, 5, 7, 10, 12

- H -

Hardware requirements, 2, 47
HOLD files, 2, 3, 4, 34

- I -

I/O character testing modes, 22
I/O operations, 1, 2, 15, 18, 19, 20, 21, 22, 23
I/O processor (see System I/O processor)
I/O specifications, 5
Identification, 3
Illegal strings, 26
Implicit calls, 7, 30
Implicit call processor, 2, 26, 27
INCLUDE control message, 7, 10, 16
Input control messages, 8
INSERT control message, 16, 17, 32
INSTALlation package, 34, 35
Interlace requirements, 2, 47
Interpretive processing, 16, 26
Interrupt level, 2, 24
Interrupt save block, 2
Interrupt service routines, 1, 8, 9, 11, 15, 24
Interruptable routines, 2, 15

- J -

JOB control message, 3

- K -

Key-in initiation, 17

- L -

LABEL control message, 5, 32
Labeled COMMON blocks, 7, 43, 46
LEAVE control message, 16, 17
Library routines, 1, 47
Line count, 19
Line printer operations, 18, 19, 20, 22
Linkage cell R\MACH, 31, 37
Listing-object option, 5, 6, 12
Listing-output file, 4, 5, 12, 13, 14, 19
Load address word, 42
LOAD control message, 4, 5, 6, 7, 8, 10, 16
Load map, 6, 7, 10
LOAD specifications, 6, 11
Loader control messages, 7
Loader control program, 2
Local variables, 2, 28, 29, 31
Logical peripheral device names, 3

- M -

Magnetic tape files, 1, 47
Magnetic tape operations, 18, 20, 21
Manual loading, 27
Memory dump program, 2, 17
Memory dumps, 1, 16, 17
Memory Protection Feature, 1
MESSAGE control message, 4
META-SYMBOL, 1, 2, 8, 12, 35, 47
META-SYMBOL I/O specifications, 5
METAXXXX control message, 5, 12
Mode indicator, 41
MONARCH compatibility, 5
MONITOR system configuration, 33
Multiple program loading, 14

- N -

Name list location word, 45

- O -

Operational labels, 3, 8
Operational table, 3, 8
Operator actions, 17, 33
Operator control messages, 8
Overlay file, 2, 26
Overlay loader, 1, 2, 10, 35
Overlays, 7, 10, 13

- P -

Padding of partial words, 22
Page ejects, 19
Page headings, 4, 19
Page numbers, 4, 19
Paper tape operations, 18, 20, 21, 22
PAUSE control message, 5, 8, 17, 24
Pause routine (see System pause routine)
Permanent files (see HOLD files)
Physical peripheral device names, 3
Pointers, 2, 18, 19, 23
Postmortem dump program, 17
POT word, 19
Primary library, 1, 2, 7, 10, 16
Primary library file, 26, 35
Processor control messages, 5
Processor file, 26, 35

Processors, 1, 17, 33, 35
Program segments (see Segments)
Programmed Operators, 41, 44
Protected files, 1, 21
Protected routines, 15, 28, 29, 31
Push-down lists, 2

- R -

RAD Files, 1, 18, 20, 23, 33, 34, 47
Random access RAD Files, 1, 18, 19, 23
Real-time operations, 1, 11, 15, 19, 24
Receiving sequences (see Standard receiving sequences)
Record address, 18, 19
Record control words, 23
RECURSIVE declaration, 15, 29
Reentrance chain, 2, 17
Reentrance monitor, 2
Reentrant routines, 2, 15, 19, 29, 31
RELEASE control message, 3, 4
Relocatable programs, 2, 6, 46
RENTAB control message, 17
REPLACE control message, 32
Reserved files, 1, 3, 4
Resident I/O drivers, 34, 35
Resident loader, 2, 11, 26
Resident loader control routine, 26
Resident monitor, 1
Resident routines, 1, 7, 11, 24
Resident symbol table, 17, 25, 26, 38
Resident symbol table search routine, 25
Resident user's programs (see User's programs)
REWIND control message, 8
Rewind operation, 8, 19

- S -

S cards, 6
SCAN control message, 32
Scanning control messages, 24
Scanning files, 18, 20, 21, 32
Scratch files, 1, 5
Secondary library, 1, 2, 6, 7, 11
Secondary library file, 26, 35
Sector address, 18, 19, 23
SEG control message, 7, 10
Segments, 2, 7, 10
Semiabsolute loader, 2, 26
Sequential RAD Files, 1, 8, 19, 20, 23
Sequential file subcontrol messages, 8
SKIP control message, 32
SNAP control message, 16
Snapshot dump, 2, 16
Spacing magnetic tape, 21
Standard assignments (see System standard assignments)
Standard calling sequences, 27, 30, 31
Standard receiving sequences, 28, 29, 30, 31
Stop-character code, 22
Subroutines with a variable number of arguments, 30
Subroutines with no arguments, 31
SYMBOL, 1, 2, 5, 6, 15
Symbol table (see Resident symbol table)
Symbol table search routine, 9
Symbolic corrections, 12, 13, 14, 47
Symbolic file names, 3, 32
Symbolic input, 5, 6, 8
Symbolic unit names (see Physical peripheral device names)
SYMTAB control message, 17
System abort routine, 8, 17, 24
System BCD-to-binary conversion routine, 9, 26
System control messages, 3
SYSTEM-DEVICE designation, 33, 34, 35, 36, 37, 39
System exit routine, 17, 24
System files, 2, 26
System generation routine, 32, 33
System generation messages, 35, 36, 37
System I/O processor, 1, 2, 19
System labels, 4,
System pause routine, 17, 24
System reserved names, 9
System standard assignments, 34, 39
System UPDATE processor, 1, 2, 32

- T -

Temporary storage locations, (see Temps)
Temps, 2, 19, 28, 29, 30, 31
TITLE control message, 4
TRACE control message, 16
Transfer word, 45
TRAP control message, 16
Typewriter operations, 18, 20, 21, 22

- U -

Undefined external symbols (see Implicit calls)
Unit Availability Table (UAT), 34, 37, 38, 39
Unit designation, 18, 19
Unit Name Table (UNT), 34, 37, 38, 39
Unrecognized control messages, 9
UPDATE processor control messages, 32
Update program (see System UPDATE processor)
User-defined control messages, 9
User's files, 1
User's programs, 1, 19, 20, 21, 22, 26
Utility routines, 2

- X -

X cards, 6

SCIENTIFIC DATA SYSTEMS 1649 Seventeenth Street • Santa Monica, California • Phone (213) UP 1-0960

SALES OFFICES

EASTERN

Maryland Engineering Center
12150 Parklawn Drive
Rockville, Maryland
(301) 933-5900

69 Hickory Drive
Waltham, Massachusetts
(617) 899-4700

1301 Avenue of the Americas
New York City, New York
(212) 765-1230

One Bala Avenue Building
Bala-Cynwyd, Pennsylvania
(215) 667-4944

SOUTHERN

Holiday Office Center
3322 South Memorial Parkway
Huntsville, Alabama
(205) 881-5746

1325 North Atlantic Avenue
Cocoa Beach, Florida
(305) 784-1555

6434 Maple Avenue
Dallas, Texas
(214) 357-0451

3334 Richmond Avenue
Houston, Texas
(713) 526-2693

MIDWEST

3150 Des Plaines Avenue
Des Plaines, Illinois
(312) 824-8147

17500 W. Eight Mile Road
Southfield, Michigan
(313) 353-7360

Suite 222, Kimberly Building
2510 South Brentwood Blvd.
St. Louis, Missouri
(314) 968-0250

One Parkway Center
875 Greentree Road
Pittsburgh, Pennsylvania
(412) 921-3640

WESTERN

1360 So. Anaheim Blvd.
Anaheim, California
(213) 865-5293 (F.X.)
(714) 774-0461 (Local)

2526 Broadway Avenue
Santa Monica, California
(213) 870-5862

Sunnyvale Office Center
505 West Olive Avenue
Sunnyvale, California
(408) 736-9193

World Savings Building
1111 South Colorado Blvd.
Denver, Colorado
(303) 756-8505

Fountain Professional Building
9000 Menaul Blvd., N. E.
Albuquerque, New Mexico
(505) 298-7683

Suite 100, Redwood Bldg.
845 106th Street, N. E.
Bellevue, Washington
(206) 454-3991

CANADA

864 Lady Ellen Place
Ottawa 3, Ontario
(613) 722-3242

FOREIGN REPRESENTATIVES

AUSTRALIA

GEC Australia Pty. Limited
GPO Box 1594
104-114 Clarence Street
Sydney, NSW, Australia

ENGLAND

International Systems Control Limited
East Lane
Wembley
Middlesex, England

FRANCE

CITEC
101 Boulevard Murat
Paris 16, France

JAPAN

F. Kanematsu & Co. Inc.
Central P. O. Box 141
New Kaijo Building
Marunouchi
Tokyo, Japan

90 11 08A