

# 3600

## COMPUTER SYSTEM SCOPE

REFERENCE MANUAL

**CONTROL DATA**  
CORPORATION

# 3600

COMPUTER SYSTEM  
SCOPE

REFERENCE MANUAL

**CONTROL DATA**  
CORPORATION



# CONTENTS

---

CHAPTER 1	SCOPE OPERATING SYSTEM	1-1
	1.1 Control Statements	1-2
	1.2 Requests	1-2
	1.3 Jobs	1-2
	1.4 Logical Units	1-2
	1.5 Minimum Unit Configuration	1-5
	1.6 Equipment Assignment	1-5
	1.7 Satellite	1-6
	1.8 Tape Labeling	1-6
CHAPTER 2	SCOPE CONTROL STATEMENTS	2-1
	2.1 SEQUENCE Statement	2-1
	2.2 JOB Statement	2-2
	2.3 FILE Statement	2-2
	2.4 EQUIP Statement	2-3
	2.5 Loading, Executing Library Programs	2-12
	2.6 Library Statement	2-21
	2.7 AET Statement	2-22
	2.8 END REEL Statement	2-23
	2.9 ENDScope Statement	2-24
	2.10 Loading Object Programs	2-24
	2.11 Executing Object Programs	2-26
	2.12 Examples of Deck Structure	2-30

CHAPTER 3	PROGRAMMER REQUESTS	3-1
	3.1 Input/Output Requests	3-1
	3.2 Stacking of Requests	3-12
	3.3 External Interrupt Control	3-14
	3.4 Internal Interrupt Control	3-14
	3.5 Clock Interrupt	3-19
	3.6 Special Requests	3-21
CHAPTER 4	DEBUGGING AIDS	4-1
	4.1 Snap Dump	4-1
	4.2 Trace Dump	4-3
	4.3 Recovery Dump	4-5
	4.4 Memory Map	4-5
CHAPTER 5	LOADER	5-1
	5.1 Loader Operations	5-1
	5.2 Loader Control Cards	5-3
	5.3 Loader Calls	5-7
	5.4 Program Assignment	5-10
	5.5 Correcting Subprograms	5-13
	5.6 Binary Card Formats	5-16
CHAPTER 6	PREPARATION OF OVERLAY TAPES	6-1
	6.1 Loader Control Statements	6-1
	6.2 Executing Overlay Programs	6-9
	6.3 Loading Overlays and Segments	6-13
	6.4 Format of Overlay Tapes	6-19

CHAPTER 7	LIBRARY PREPARATION AND MAINTENANCE	7-1
	7.1 Input/Output	7-2
	7.2 Record Types	7-3
	7.3 Control Statements on Standard Input Unit	7-7
	7.4 Changing PRELIB	7-19
	7.5 Table of Contents	7-20
SECTION A	AVAILABLE EQUIPMENT TABLE	A-1
SECTION B	MACRO DEFINITIONS AND CALLING SEQUENCES	B-1
	Internal	B-1
	Input/Output	B-2
	Interrupt	B-7
	Special	B-9
SECTION C	SCOPE MESSAGES AND DIAGNOSTICS	C-1
	Messages on OUT	C-1
	Messages on ICM	C-15
	Messages on OCM	C-18
	Messages on ACC	C-20
	Messages on PUN	C-21
	Diagnostics on OUT	C-21
	Diagnostics on OCM	C-30

---

The SCOPE monitor system for the Control Data<sup>®</sup> 3600 computer facilitates job processing and simplifies programming and operating by providing:

- Job Processing

- Assigns equipment

- Initiates compilations and assemblies

- Loads, links and initiates execution of subprograms

- Allocates storage

- Provides OVERLAY processing and accounting information

- Communicates with Satellites<sup>®</sup>  
for input/output tape handling

- Debugging Aids

- Diagnostics

- Octal corrections

- Special debugging dumps

- SNAP

- TRACE

- Memory map

- Error dumps (recovery dumps)

- Input/output Control and Special Requests

- Input/output routines with drivers

- External interrupt control

- Tape handling - including labeling and continuation reels

- Internal interrupt control

- Sampling of time, date, equipment status, and available storage

- Library Preparation and Maintenance

- Preparation of a new library tape

- Editing

- Listing the table of contents

Monitor operations are specified in a job by control statements and programmer requests.

The resident section of SCOPE occupies the low numbered locations of bank zero. Resident includes the control routine, EXEC, input/output control, IOC, interrupt control, INC, equipment assignment, EQA, and the Satellite control program, SCP, if used. The loader is adjacent to resident in lower storage. System I/O drivers and the library table of contents are loaded into the highest locations of the highest available bank. The system I/O drivers and SCOPE are protected by an internal interrupt system.

## 1.1

### **CONTROL STATEMENTS**

SCOPE control statements have the following card format: a 7,9 punch in column 1; a statement name (beginning in column 2); and parameters, if required, separated by commas.

Control statements are free-field, but must be contained on a single 80-column card. No terminating character is needed.

## 1.2

### **REQUESTS**

A request is written as a macro instruction and assembled into a calling sequence to a SCOPE routine. Requests may be written as calling sequences in assembly language programs.

The request name begins in column 10 and is terminated by a blank column. The address field may begin anywhere after the blank and before column 41. Unless noted otherwise, requests may contain any legal COMPASS expression in the address field. Further details on statement parameters are given in the COMPASS Reference Manual.

## 1.3

### **JOB**

A job includes all operations indicated between JOB cards or between a JOB statement and an ENDScope statement. Each job is terminated with an end-of-file mark. A job may consist of multiple assemblies, compilations, and executions.

A job stack consists of a group of jobs placed together for processing by the monitor control system. The stack begins with the first monitor statement and ends with an ENDScope statement.

## 1.4

### **LOGICAL UNITS**

In SCOPE control statements and requests, the logical unit designation allows reference to a piece of equipment within the logic of the program; it is



independent of the physical unit designation. The programmer designates input and output units with logical unit numbers 1 through 79. Mnemonics may be used in programmer requests. Logical units are divided into three classes:

PROGRAMMER UNITS (logical units 1-49)

These units are assigned throughout the job for reference by the program. When a tape is released at the end of the job, it may be unloaded and saved for the programmer, if specified; or made available for reuse in a later job.

Mnemonic

SCRATCH UNITS (logical units 50-59)

S0, S1, . . . S9

Scratch units may be referenced at any time by the programmer, but they are released after each execution and may not be saved.

SYSTEM UNITS (logical units 60-80)

The system units assigned by the monitor system are used by SCOPE and the programmer and, with two exceptions, are released only at the end of the job stack. The load-and-go unit, available to compilers and assemblers, is released at run time; auxiliary libraries are released at the end of each job.

	Logical Number	Mnemonic
STANDARD INPUT	60	INP
The control cards for all SCOPE jobs are placed on this unit by the operator. Frequently, the programs and data to be processed are also on INP.		
STANDARD OUTPUT	61	OUT
SCOPE control statements, diagnostics, dumps, and loader control cards are written in BCD mode on this unit. Program output may also be written on OUT.		
STANDARD PUNCH OUTPUT	62	PUN
Program and SCOPE output for punching is recorded on this unit.		
INPUT COMMENT	63	ICM
Comments from the operator to the monitor system are made on this unit. The programmer may also use ICM for input directions.		

	Logical Number	Mnemonic
OUTPUT COMMENT	64	OCM
<p>Statements from the monitor system to the operator are made on this unit. The programmer may also list information on this unit. ICM and OCM are usually assigned to the console typewriter. OCM must be assigned and operable, otherwise the jobs are terminated.</p>		
ACCOUNTING INFORMATION	65	ACC
<p>Job statements, date, time on and off, and elapsed job time are written on this unit for installation records.</p>		
SATELLITES	66-68	none
<p>Satellite units are used by the system when it is operating in Satellite mode. SCOPE contains a control program for handling Satellite input/output requests. Programmers cannot control or reference the Satellite units.</p>		
LOAD-AND-GO	69	LGO
<p>Binary object programs transferred from the standard input unit or produced by compilation or assembly may be stored here prior to loading and executing. This unit may be saved by the programmer with an EQUIP statement. If saved, it will be released at the end of the job. If not saved, it will be released at the beginning of the run.</p>		
SCOPE LIBRARY	70	LIB
<p>The SCOPE library contains the monitor system and all programs and subroutines which operate under SCOPE, such as, FORTRAN, COBOL, COMPASS, SORT, and ALGOL.</p>		
AUXILIARY LIBRARIES	71-79	none
<p>Auxiliary libraries are used for library preparation and editing and additional system libraries.</p>		
SYSTEM SCRATCH RECORD	80	SCR
<p>Monitor equipment tables, accounting data, and recovery dump information for each job are stored in the first record of this unit. If part of resident is destroyed by a running program, the system uses the information on this tape</p>		

for recovery. The first scratch unit requested by a program is made equivalent to this unit. The programmer may never reference logical unit 80.

An available tape must be provided on a unit numbered 1 through 7 for assignment to SCR at autoload time. Since the number is set in the STOP switches, the STOP switches are not available for programmer use.

## 1.5 MINIMUM UNIT CONFIGURATION

The minimum configuration required by SCOPE includes:

INP	ACC (may be bypassed)
OUT	LIB
ICM	SCR
OCM	

Compilers and assemblers operating under SCOPE may require additional units.

## 1.6 EQUIPMENT ASSIGNMENT

SCOPE assumes that all programmer and scratch logical unit numbers refer to unique, high-density (556bpi), labeled, magnetic tapes read or written in binary mode; and that tapes are not saved at the end of a job. If programmer and scratch units meet this description, no special equipment declarations are needed.

The normal equipment assignments may be altered by:

### EQUIP

- Equate a logical unit to a specific hardware type
- Equate different logical units to a single physical unit
- Declare a tape to be unlabeled

### EQUIP or LABEL

- Indicate a desired tape label for a logical unit

### MODE

- Indicate the recording mode of a tape unit

**EQUIP or MODE**

Select a particular density for a tape unit

Restrict the use of a unit

**EQUIP or SAVE**

Indicate that a tape is to be saved at the end of the job

**1.7**

**SATELLITES**

Satellite computers may be used in conjunction with the 3600 computer. The 3600 computer is set to Satellite mode when Satellites are performing peripheral processing, input tape preparation, output tape processing or accounting for these jobs through 3600 communication.

**1.8**

**TAPE HANDLING**

The label provided by SCOPE or defined by the programmer is the first record of a tape. Tapes handled by SCOPE are usually labeled. An unlabeled tape must be declared, and its physical unit number fixed. A logical tape may contain any number of reels, the labels of which differ from one another only in reel number. Reels of the same tape are usually written in the order 1, 2, 3, . . . , 99; reel 1 follows reel 99.

<u>Character Position</u>	<u>Parameters</u>
1	Density 2, 5 or 8
2-3	Unique label identifier ( )
4-5	Logical unit number - 2 BCD digits
6-8	Retention code - 3 BCD digits
9-22	File name - 14 alphanumeric characters
23-24	Reel number - 2 BCD digits
25-30	Date written - month, day, year in BCD (mmddy)
31-32	Edition number - 2 BCD digits
33-80	Any information user chooses

**MODE**

All labels, except SCR and LIB, are read and written in BCD mode (even parity).

## WRITE ASSIGNMENT

When a tape is to be assigned for writing, each tape with a write enable ring is examined by SCOPE. In examining the first record, SCOPE determines if the label is standard by checking character positions 2 and 3 for left and right parentheses. If the record is a standard label, SCOPE adds the retention code and the date written and compares the result to the current date. The tape is considered available if the current date is greater than the result. If the record is not a label the tape is considered available. When a tape is needed for output, SCOPE searches for an available tape. If a tape cannot be found, SCOPE notifies the operator who may remedy the situation as indicated in Appendix C, Messages on ICM and Messages on OCM. When a tape is assigned, SCOPE types the label identification (positions 9-32) for that unit.

Retention code is tested twice during a job, when tapes are assigned and when they are released. When the retention code is zero or blank, the tape is not to be saved. A retention code of 1 through 998 specifies that number of days; 999 reserves the tape permanently.

## LABEL WRITING

The programmer supplies the data for the first four words of a label with either a LABEL request or an EQUIP statement. The WRLABEL instruction specifies the last six words of the label as user supplied information and transmits the full ten words from core storage to the magnetic tape.

### Name or

### Logical Unit

Whenever a label is written, either a name or logical unit is written in the label. If both are specified, the name is written. The name may be specified in either an EQUIP statement or a LABEL request. The logical unit is either the master unit (specified in WRLABEL or the first output request) or the unit specified in an EQUIP statement or LABEL request. The file name is 14 alphanumeric characters. If the file name is blank, the tape will be a numbered tape; edition, retention code or date written will be the same as that specified on a label request for that logical unit.

### Identification

Label identifiers ( ) are always written in the label.

### Density

Density designation is always written in the label.

### Reel Number

Declaration of reel number is optional. It may be specified in either an EQUIP statement or LABEL request. If it is not specified for output reels, SCOPE assumes the first reel is number 1 and increments by 1 for each subsequent reel. If previous reels are numbered and no reel number is specified for continuation reels, the last reel number is incremented by one.

Edition Number	Declaration of edition number is optional. It may be specified in an EQUIP or LABEL request. If it is not specified, SCOPE writes blanks for edition number.  A specified edition number is the same for all reels of a file. If edition is not specified for continuation reels, SCOPE assigns the edition number of previous reels.
Retention Code	Retention code may be specified in an EQUIP statement or LABEL request. If not specified, zero is used but the tape may be saved via the SAVE request or EQUIP statement and unloaded at the end of the job. When the retention code is non-zero, a SAVE request for that unit is not needed.
Date	The date written is specified by the operator at initial autoloading and written in each label.
User's Area	Information may be specified in a WRLABEL request, otherwise BCD zeros are written in these 48 character positions.

**LABEL READING** A LABEL request or an EQUIP statement specifies the first four words of data to be interrogated in an input label. The programmer can access the last six words via the RDLABEL instruction which transmits ten words from the magnetic tape to core storage.

Name or Logical Unit	When labels are used for assigning logical units, either the unit number or the label name is checked. If specified in an EQUIP statement or LABEL request, the name is used. A logical unit differing from that specified in the call may be specified in EQUIP or LABEL.
Identification	The label identifiers and the record length are used to determine whether the first record on tape is a label.
Density	The density character specifies the recording density for reading and/or checking this label.
Reel Number	Reel number is optional and may be specified with a LABEL request or EQUIP statement. Unless otherwise specified SCOPE reads the lowest numbered input reel first and increments the number by one for each subsequent reel. For continuation reels, a number one greater than the previous reel is sought.

Edition Number	The edition number, which is the same for all reels of a file, may be specified in an EQUIP or LABEL request. If no edition number is specified, the field is ignored when checking labels and making the assignment.
Retention Code	When input reels are sought, retention code is ignored.
Date	The date specified in an EQUIP or LABEL statement must exactly match the date on the label. If a date is not specified, the date on the label is ignored.
User's Area	Information in this area may be interrogated via the RDLABEL request. If the first request on a unit is not RDLABEL, this information is ignored.

## ASSIGNMENT PROBLEMS

No available input reel, duplicate input reels, (except on RDLABEL) no available reel for output, or non-standard label assignment (unlabeled tapes) may prevent the normal assignment of units. The operator is queried via the OCM unit and must reply via the ICM unit. For specific detail, see Appendix C, ICM messages and OCM messages.

### **Example:**

Write on logical unit 25, high density magnetic tape, in binary mode, with standard labeling. Without an EQUIP statement or LABEL request, SCOPE searches for an available tape and writes a label containing:

density	5
unique label identifier	( )
logical unit number	25
retention code	000
file name (14 characters)	blank
reel number	MMDDYY entered by operator at beginning of SCOPE run
edition number	2 BCD blanks
6 words	BCD zeros

If a tape without a label is to be used, it must be declared as unlabeled. If a name or a unit number different from that appearing in the request is to be read or written in a tape label, the name or number must be declared. All labels contain either a name or a logical unit number; library tape labels have both.

**LABEL SKIPPING** If a unit is positioned at load point when a request which produces tape motion is issued, the label will be skipped by the system. In the case of RDLABEL and WRLABEL, the skipping is not performed.

---

Jobs and directives are submitted to SCOPE on the standard input unit. Jobs are executed in sequence from the standard input unit or as specified by the operator.

## 2.1 SEQUENCE STATEMENT

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ SEQUENCE,n

This statement assigns a job sequence number, n, which serves as identification for scheduling purposes. SCOPE does not check the order of the numbers.

The SEQUENCE statement immediately precedes a JOB statement on the standard input unit. An end-of-file mark should precede each SEQUENCE statement (section 2.10).

When the SEQUENCE statement is encountered, SCOPE unloads saved tapes, and releases all other programmer units, scratch units, and the load-and-go unit. The current job is terminated and the new job initiated.

### **Example:**

```
<end-of-file>
 $\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ SEQUENCE, 4
 $\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ JOB, . . .
.
.
.
end-of-file
 $\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ SEQUENCE,5
 $\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ JOB, . . .
```

In Satellite mode, SEQUENCE cards are recognized but are renumbered by the Satellite computer.



## 2.2

### JOB STATEMENT

All programs submitted for processing under SCOPE start with a JOB card which signals the beginning of a job, provides accounting information for the installation, identifies the programmer, and sets a job processing time limit.

<sup>7</sup>JOB,c,i,t  
<sub>9</sub>

- c the charge number; unlimited number of alphanumeric characters. SCOPE truncates to 8 characters in Satellite mode for Satellite accounting.
- i the programmer identification. It may be any length and appears as given in the control card listing; it is truncated to 6 characters for operator identification.
- t the maximum time limit in minutes allowed for the entire job including operator functions. No job may exceed 2184 minutes; if a limit is not specified, the maximum is assumed.

The job is terminated if the c and i fields are not present. A single JOB card may be used for any number of independent programs; however, the time specified is the maximum allowed for the combined programs.

## 2.3

### FILE STATEMENT

Binary records can be transferred from the standard input unit to another logical unit by the FILE statement.

<sup>7</sup>FILE,u  
<sub>9</sub>

All records which follow the FILE statement up to a FILE END statement are transferred to logical unit u:1-59, or 69 (load-and-go). The records are written in odd (binary) parity.

<sup>7</sup>FILE END  
<sub>9</sub>

SCOPE writes an end-of-file mark and then backspaces over it when the FILE END statement is encountered. Any number of FILE - FILE END sequences may be directed to the same or a different logical unit. Any type of binary data may be transferred with a FILE statement except the SCOPE control statements: SEQUENCE, JOB, END REEL, END SCOPE. An attempt to transfer any of these statements will terminate the job.

## 2.4 EQUIP STATEMENT

Normally, the EQUIP statement precedes the RUN card of the program or the entry point name statement. However, it may be entered on Input Comment Unit by the operator during a run. EQUIP statements are required to declare programmer or scratch units which are not unique, standard-density, labeled, magnetic tapes.

<sup>7</sup>  
<sup>9</sup>EQUIP, u=d<sub>1</sub>, d<sub>2</sub>, d<sub>3</sub>, . . . , d<sub>n</sub>

u logical unit number; 1-59, 69, 71-79 from INP;  
1-65, 69-79 from ICM

d declaration

The types of EQUIP declarations are:

hardware	disposition	assignment
usage	equivalence	
density	labels	

The logical unit category determines how long the EQUIP declaration remains in force. Declarations pertaining to scratch units are in effect for a single execution only. Programmer unit declarations carry over between executions of a job, unless changed by other EQUIP statements. System unit declarations remain until all jobs have been processed, unless specifically redefined.

Declarations may be combined in one EQUIP statement; however, there is no error checking. If they conflict, the last declaration in the list takes precedence.

## HARDWARE DECLARATIONS

<sup>7</sup>  
<sup>9</sup>EQUIP, u=hhn

hh specifies the hardware type:

MT	magnetic tape	LP	on-line printer
CR	card reader	DR	drum
PT	paper tape station	TY	typewriter
CP	card punch	TV	display
DF	disk file	PL	plotter

n is the AET ordinal of the hardware type, (Appendix A). If n is absent, the next available physical unit of the hardware type is assumed to be referenced.

When hh specifies a Satellite, SA, SB, SC, SD, SE, or SF, n is the octal ordinal for the unit connected to the specified Satellite. It must be specified.

After an EQUIP statement is processed, SCOPE types a message indicating which physical unit has been assigned. (Appendix C, Messages on OCM).

**Examples:**

```
7
9EQUIP,10=CR
```

The first available card reader is assigned to unit 10.

```
7
9EQUIP,12=CR2
```

Logical unit 12 will be the second card reader in the Available Equipment Table.

```
7
9JOB,3215079,PETE,15
```

```
7
9EQUIP,10-CR2
```

```
7
9EQUIP,9=CP1
```

object program

```
7
9RUN,7,300,7,1
```

< end-of-file >

```
7
9EQUIP,11=CR3
```

object program

```
7
9RUN,6,250
```

< end-of-file >

Logical unit 10 is the second card reader, logical unit 9 is the first card punch, and logical unit 11 is the third card reader in the AET.

USAGE  
DECLARATIONS

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ EQUIP, u=hh

hh RW read/write; all operations are allowed.

BY bypass; all references to this unit except MODE (usage) and STATUS are treated as no operation.

RO read only; output operations (WRITE, WEOT, WRLABEL, MARKEF and ERASE) cause job to be abandoned.

Read-only units such as card readers need not be declared as such.

Usage declarations may be made for logical units 1-59, 65, 71-79.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ EQUIP, 11=RO Unit 11 is a read-only unit.

DENSITY  
DECLARATIONS

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ EQUIP, u=hh

hh LO low density magnetic tape (200bpi)

HI high density magnetic tape (556bpi)

HY hyper density magnetic tape (800bpi) or the highest density of which the unit is capable.

Density declarations are allowed for any acceptable logical unit not yet referenced.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ EQUIP, 12=HY

Density declarations may be followed by a magnetic tape ordinal, n.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ EQUIP, 5=HI05 processed as if the statement were:

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ EQUIP, 5=MT05, HI

**ASSIGNMENT  
DECLARATION**

<sup>7</sup><sub>9</sub>EQUIP, u=DA (units 1-49 and 71-79)

DA Defer automatic assignment of logical unit; SCOPE requests assignments from the operator.

**DISPOSITION  
DECLARATIONS**

<sup>7</sup><sub>9</sub>EQUIP, u=hh

hh SV save tape at end of job. SCOPE unloads tape at end of job and directs a message to the operator. All tapes not saved become available for reuse when the limit set by the retention code expires. Only logical units 1-49, 69-79 may be saved.

PR print tape at end of job.

PU punch tape at end of job.

PP print BCD records and punch binary records at end of job.

More than one disposition declaration may appear in an EQUIP statement, but only the last will be effective. SV, PR is the same as PR, SV. PU, PR is not the same as PP; only PR would be effective. PR, PU and PP declarations are allowed for logical units 1-49 and 69.

EQUIP STATEMENTS MAY ORIGINATE FROM:		LOGICAL UNITS	THESE EQUIP DECLARATIONS ARE LIMITED TO THE CHECKED UNITS					
INP (60)	ICM (63)		USAGE EQUIP, u = RW BY RO	LABEL EQUIP, u = ( )	DISPOSITION		EQUIVALENCE EQUIP, u = u'	
					EQUIP, u = SV	EQUIP, u = PR, PU, PP	u	u'
YES	YES	1-49	YES	YES	YES	YES	YES	YES
YES	YES	50-59	YES				YES	YES
	YES	60, 63						YES
	YES	61, 62, 64						YES
	YES	65	YES					
		66-68						
YES	YES	69			YES	YES		YES
	YES	70			YES			
YES	YES	71-79	YES	YES	YES		YES	YES
		80						
DECLARATIONS CONCERNING HARDWARE TYPE AND DENSITY MAY BE MADE FOR ANY ACCEPTABLE UNIT.				ANY ACCEPTABLE UNIT MAY BE DECLARED TO BE UNLABELLED EQUIP, u = **			EQUIVALENCING OF SYSTEM UNITS IS LIMITED TO: 62 = 61 65 = 61 65 = 62	

Satellite mode:

PR, PU, PP cause the tape to be turned over to the Satellite for processing. The tape is blank labeled and returned to the system unless SV is specified, in which case, the tape is unloaded after being processed by the Satellite.

Non-Satellite mode:

PR, PU, PP cause the tape to be unloaded; a message is typed to the operator.

<sup>7</sup><sub>9</sub>EQUIP, 2=SV, PR

In Satellite mode, logical unit 2 is printed by the Satellite, and unloaded.

In non-Satellite mode, a PRINT message is given to the operator on OCM and logical unit 2 is unloaded.

## EQUIVALENCE DECLARATIONS

### Master Logical Unit

<sup>7</sup><sub>9</sub>EQUIP, u=u'

u' unit is made equivalent to u.

Unit u must be unassigned when the declaration is made, unless it is a system unit. Units 1-59, 71-79 (u) may be equated to units 1-64, 69, 71-79 (u'). System unit equivalencing is restricted to the following:

62 = 61

65 = 61

65 = 62

If two logical units are equivalent they refer to the same physical unit. Any number of units may be equated to each other, but separate EQUIP statements must be used for each pair. Only one u' may be equated to one u per EQUIP card. The master logical unit is the last unit in the string of equivalences.

**Examples:**

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 30=61$	unit 30 is equated to standard output unit (61)
$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 31=32$	units 31 and 32 are the same physical unit
$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 2=6$	} unit 6 is the master logical unit
$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 3=2$	
$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 1=2$	} unit 4 is the master logical unit
$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 2=3$	
$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 3=4$	
$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 1=2$	} unit 2 is the master unit
$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 3=2$	

**UNLABELED TAPE  
DECLARATION**

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, u=**$

\*\* unlabeled or non-standard labeled tape.

Any acceptable logical tape unit may be declared unlabeled.

**READ** The programmer must inform the operator which physical reel should be mounted. With an  $\text{EQUIP}, u=hhnn$ , the operator must define the specific physical tape unit containing the reel.

**WRITE** SCOPE looks for an available unit.

When unlabeled tapes are used, the first read command will read the first record on the tape. When tapes are labeled, the first read command reads the second record on the tape.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix} \text{EQUIP}, 20=**, HI, SV$

This statement makes it possible to read or write on an unlabeled, high density tape (556 bpi), unit 20. The SV (save) declaration causes SCOPE to unload the tape at the end of the job.

<sup>7</sup>  
<sup>9</sup>EQUIP,20=\*\*,MT

SCOPE attempts to locate an available tape. Therefore, the MT form of the EQUIP statement should not be used for an input tape.

**LABELED TAPE  
DECLARATIONS**

<sup>7</sup>  
<sup>9</sup>EQUIP,u=(name, edition, reel, retention code or date)

Logical units 1-49, 71-79 may be labeled.

Name may contain up to 14 alphanumeric characters. Name begins with the first character after the left parenthesis and ends with the comma or right parenthesis; blanks will be filled in to the right.

Edition and reel number (1 to 99) are optional; but they should be used if two or more labels have the same name. If no reel number is specified, the lowest reel number is taken for input and the output reel is numbered 1.

If logical unit u represents an output tape, the last parameter is the retention code; a blank or zero specifies that the tape is not to be retained. A code of 1 through 998 indicates the number of days the tape is to be saved; 999 indicates the tape is to be retained indefinitely.

If logical unit u represents an input tape, the last parameter is the date the tape was originally prepared. If a date is specified, SCOPE compares this date with the date contained in the label. If no date is specified, no check is performed.

**Examples:**

<sup>7</sup>  
<sup>9</sup>EQUIP,25=(INVENTORY,1,1),LO,RO,SV

Logical unit 25 contains a tape labeled INVENTORY, edition 1, reel 1. It is a low density, read-only unit, which is to be unloaded at the end of the job. The date is not to be checked.

<sup>7</sup>  
<sup>9</sup>EQUIP,10=(RUSSIAN-ENGLISH LEXICON,,1,999)

Label a tape (logical unit 10) with RUSSIAN-ENGLISH LEXICON reel 1, the first operation on this tape is a write. The tape is to be permanently reserved.



**LOGICAL  
UNIT NUMBER  
DECLARATIONS**

<sup>7</sup><sub>9</sub>EQUIP, u>(\*nn, edition, reel, retention code or date)

Logical units 1-49, 71-79 may be labeled.

nn is the 2-digit logical unit number, 1-79, contained in the tape label (this is not the tape name) or to be written in the label. If no tape with nn in its label is found when the first request for unit u is a READ request, SCOPE will ask the operator to supply the tape or terminate the job. If the first request for unit u is a WRITE request, the number nn will be written in the label.

Only u may be used in a request for that unit. If the programmer wishes to use a request with nn, he must first give EQUIP, u=nn.

<sup>7</sup><sub>9</sub>EQUIP, 28>(\*35, 1, 1, 30)

The tape assigned to logical unit 28 contains 35 as the logical unit number in the tape label, although it is referenced as unit 28 in the program.

This statement may be used to write a tape referenced as logical unit 28 with unit number 35 in the label. It will be retained for 30 days from the current date.

EQUIP statements precede the program in which the logical unit is referenced. In the following example, FILE A, edition number 1, is the name of unit 20. CHANGE FILE is the name of unit 21. Unit 22 has the same name as unit 20, but it is the second edition. All units are to be saved.

<sup>7</sup><sub>9</sub>JOB, 20, BETA 15

<sup>7</sup><sub>9</sub>EQUIP, 10=CR

<sup>7</sup><sub>9</sub>EQUIP, 11=CP

<sup>7</sup><sub>9</sub>EQUIP, 20=(FILE A, 1), SV, RO, LO

<sup>7</sup><sub>9</sub>EQUIP, 21=(CHANGE FILE), RO, SV, LO

<sup>7</sup><sub>9</sub>EQUIP, 22=(FILE A, 2), SV, LO

<sup>7</sup><sub>9</sub>LOAD, 49

<sup>7</sup><sub>9</sub>RUN, 13, 3000

In the next example, high density (556 bpi) logical units 5 and 20 are to be saved. COSY input to the COMPASS assembly is on logical unit 5. COSY output is to be on logical unit 20.

```
7
9JOB 30, QWERTY, 30
```

```
7
9EQUIP, 5=(COSY TAPE, 1, 1, 20565)
```

```
7
9EQUIP, 20=(COSY TAPE, 21, 1, 999)
```

```
7
9COMPASS, Y=5, C=20, L, X
```

```
IDENT
```

```
.
```

```
.
```

```
.
```

```
SCOPE
```

```
7
9EQUIP, 25=(*40)
```

```
7
9EQUIP, 26=**, HY
```

```
7
9LOAD
```

```
7
9RUN, 28, 1000
```

Logical unit 25 contains a tape with a label bearing the logical unit number 40; input/output requests within the program which reference 25 will reference this tape. (If 40 were referenced, a new unit would be assigned; both labels would contain the unit number 40.)

The hyper density (800 bpi) tape on unit 26 has a non-standard label. The physical unit must be specified by the operator via ICM when the unit is read.

2.5  
**LOADING,  
EXECUTING  
LIBRARY  
PROGRAMS**

Library programs<sup>†</sup> are referenced by statements which name the entry points to them (FTN, COMPASS, ALGO, ALDAP, COBOL). Execution of this statement directs that the referenced program be loaded. Control is then given to the program and the operation is performed. Upon return of control to SCOPE, the lower address of the A register indicates whether errors were (non-zero) or were not (zero) detected during compilation or assembly. The upper address portion of the A register indicates whether the library subroutine must be reloaded before being re-entered by another call. If it is non-zero, the library subroutine will be reloaded, otherwise not.

<sup>7</sup><sub>9</sub>entry point name,  $p_1, p_2, \dots, p_m$

$p_i$  are parameters interpreted by the library program.

Assembly options are indicated by free-field parameters separated by commas. Parameters may appear in any order. The parameters may be followed by =n, indicating a non-standard unit is to be used.

**COMPASS** <sup>7</sup><sub>9</sub>COMPASS, assembly options

<u>Options</u>	<u>Meaning</u>
I	BCD source language input
Y	COSY input
P	Punch relocatable binary object program deck
C	COSY output
X	Write relocatable binary object program output for load-and-go
.L	List source language subprogram
B	Punch source subprograms from a COSY deck
R	Cross referenced symbol table listing
M	Produce full macro/ECHO detail list

Specific information concerning the COMPASS card is in the 3600 COMPASS Reference Manual.

---

<sup>†</sup>Library programs should not be confused with SCOPE library subroutines which are called by symbolic reference in a subprogram.

**Examples:**

$\frac{7}{9}$ COMPASS, P, X, L

Source input is on logical unit 60.

Binary deck is punched on logical unit 62.

Load-and-go tape is produced on logical unit 69.

Source language programs are listed on logical unit 61.

$\frac{7}{9}$ COMPASS, L=25, P=10

Source input is on logical unit 60.

Source language programs are listed on logical unit 61.

Binary output is punched on logical unit 10.

$\frac{7}{9}$ COMPASS, Y=1, L, C=2

BCD input is on logical unit 60.

COSY input is read from logical unit 1.

Source language programs are listed on logical unit 61.

COSY output is on logical unit 2.

**FORTRAN**

$\frac{7}{9}$ FTN, options

A terminal period is optional; the field may also be terminated by the end of the card.

<u>Options</u>	<u>Meaning</u>
L	List source language program
P	Punch relocatable binary deck
X	Write load-and-go tape
A	List assembly language program, COMPASS
I	BCD source language input
C	Punch a COSY output deck
B	BCD assembly output
*	Compile code for one bank

Specific information concerning the FTN card is in the 3600 FORTRAN Reference Manual.

**Examples:**

<sup>7</sup><sub>9</sub>FTN, I=49, L, A

Source input is on logical unit 49.

Source and compiled program are listed on unit 61.

<sup>7</sup><sub>9</sub>FTN, A, L, X

Source input is on logical unit 60.

Source and compiled program are listed on unit 61.

Binary object program is written on unit 69.

**COBOL**

<sup>7</sup><sub>9</sub>COBOL, options

The parameter list may be terminated by a blank or a period.

Options

Meaning

Z	Suppress source program listing
X	Write object program on load-and-go unit
M	Print a data map on standard output unit
P	Write relocatable binary object program on standard punch unit
T	List trivial errors
L	List the object program on standard output unit
N	Number source code lines on source listing; source input deck numbers are ignored

Specific information concerning the COBOL card is in the 3600 COBOL Reference Manual.

**Examples:**

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ COBOL, Z, X

suppress the source program listing and write object program on standard load-and-go unit.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ COBOL, X

print source program listing on OUT and write the object program on standard load-and-go unit.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ COBOL

treated as  $\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ COBOL, X.

**ALGOL**

ALGOL 60 programs may be compiled in two modes. ALGO mode compiles and executes immediately. ALDAP compiles; execution may occur immediately or later.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ ALGO

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ ALDAP, options

Options

Meaning

L	List source program
A	List assembly language program, COMPASS
P	Punch relocatable binary object program
E or X	Prepare load-and-go tape
B	Punch BCD COMPASS cards
O	Assign a unit as translator overflow tape
I	BCD source language input

Specific information concerning the ALGO and ALDAP cards is in the 3600 ALGOL Programming Systems Bulletin.

**Examples:**

<sup>7</sup><sub>9</sub>ALGO

The source program following the ALGO card on the standard input unit is compiled and executed.

<sup>7</sup><sub>9</sub>ALDAP, L, P, X

- Source input is on logical unit 60.
- Binary cards are punched on logical unit 62.
- Source program is listed on logical unit 61.
- Load-and-go tape is prepared on logical unit 69.

**SOURCE DECK  
STRUCTURE**

The entry point name statement is followed by the source program to be assembled or compiled by the library program.

A FTN card may be followed by either FORTRAN or COMPASS source language subprograms to be compiled or assembled.

A COMPASS card is followed by COMPASS subprograms to be assembled.

A COBOL card is followed by a COBOL source program to be compiled.

An ALDAP card may be followed by an ALGOL source procedure to be compiled or compiled and executed or a COMPASS subprogram to be assembled. An ALGO card is followed by an ALGOL source procedure to be compiled and executed or a COMPASS subprogram to be assembled.

If more than one library program (compilers or assemblers) is used for processing the subprograms for a single program, the source decks are stacked consecutively.

When there are consecutive calls to the same library program with no intermediate runs, the library program is not reloaded. This is true even if the calls occur in different jobs.

For each source language subprogram, a binary object program will be produced and stored on the logical unit designated by one of the parameters on the entry point name card. After each binary program is written on a logical unit, the library program writes an end-of-file mark and backspaces over it. Therefore, after all binary subprograms are written on the unit, only one end-of-file mark will remain; and the unit will be positioned so that any subsequent compilations for that unit will write over it. Consequently, at the end of the last compilation, an end-of-file mark remains on the unit.

A SCOPE card is required to indicate the end of the source subprograms following a FORTRAN or COMPASS card; also to indicate the end of the source procedures following an ALDAP card. An EOP card is required to indicate the end of the program following an ALGO card. COBOL does not require a SCOPE card. The word SCOPE begins in column 10; since it is not a SCOPE control card, there is no 7,9 punch in column 1.

**Examples:**

1	7	10	
7	COMPASS	C,L,R	
9		IDENT ANDY	
		:	COMPASS subprogram
		:	
		END	
		IDENT BRENDA	
		:	COMPASS subprogram
		:	
		END	
		SCOPE	

1	7	10	
7	FTN,A	L,P	
9		PROGRAM ONE	
		:	FORTRAN program
		:	
		END	
		IDENT TWO	
		:	COMPASS subprogram
		:	
		END	
		SUBROUTINE TWO	
		:	FORTRAN subroutine
		:	
		END	
		SCOPE	



1	10	
7	9	ALDAP, L, P, A
		PROGRAM ARCO
		:
		'EOP'
		:
		'EOP'
		SCOPE
		ALGOL program
		ALGOL procedure

1	8	
7	9	COBOL, M, L, P, T
		IDENTIFICATION DIVISION.
		:
		END PROGRAM.
		COBOL program

1	10	
7	9	ALDAP, L, P, A
		IDENT FOR
		:
		END
		:
		'EOP'
		:
		'EOP'
		IDENT TAG
		:
		END
		SCOPE
		COMPASS subprogram
		ALGOL procedure
		ALGOL procedure
		COMPASS subprogram

## DEBUGGING PARAMETERS

The parameters,  $s_1$  and  $s_2$ , in the entry point name statement are used to modify or debug the library program (not the program being compiled or assembled).

<sup>7</sup>entry point name ( $s_1, s_2$ ),  $P_1, P_2, \dots, P_m$

$s_1, s_2$  are designators C and D; they may be interchanged or either one may be absent.

C Octal correction cards for the library program are submitted on INP. The corrections apply only to the subprogram containing the specified entry point. The octal correction cards are terminated by a single TRA card. Card formats OCC and TRA are given in 5.6.

If subprograms other than those with a main entry point are to be corrected, use the CORRECT loader control card (5.2). Following the CORRECT card, are the OCC cards and 1 TRA card for each name on the CORRECT card.

- D SNAP or TRACE statements for the library program are submitted on INP. The names on the SNAP and TRACE cards may be relative to any program name, entry point, or common block used during the loading of the specified entry point name (Chapter 4).

**Examples:**

SNAP or TRACE statements only

<sup>7</sup>  
<sub>9</sub>epname (D), p<sub>1</sub>, p<sub>2</sub>

TRACE and OCC

<sup>7</sup>  
<sub>9</sub>epname (D,C), p<sub>1</sub>, p<sub>2</sub>

or

<sup>7</sup>  
<sub>9</sub>epname (C,D), p<sub>1</sub>, p<sub>2</sub>

When the SCOPE parameters are used, the OCC and TRA cards follow the entry point statement. If both C and D are used, the octal correction cards must precede SNAP and TRACE cards. A RUN statement must be included to initiate execution. Source decks to be assembled or compiled follow the RUN statement.

**Examples:**

1	10
<sup>7</sup> <sub>9</sub> COMPASS	(C), C, L, X
OCC	
OCC	
TRA	
<sup>7</sup> <sub>9</sub> RUN, 2, 400, 5	
	COMPASS Subprograms
	SCOPE

See section 5.6

1	10
7 9	FTN (D), L, X
7 9	SNAP, xxx
7 9	SNAP, xxx
7 9	TRACE, xxx
7 9	RUN, 5, 1000, 3
	FORTRAN Subprograms
	SCOPE

See chapter 4 for options

1	10
7 9	COBOL (D, C), X, L
	OCC
	OCC
	TRA
7 9	SNAP, xxx
7 9	SNAP, xxx
7 9	RUN, 2, 100, 7
	COBOL program

1	10
7 9	COMPASS (C, D), L, X
- 0	
7 9	CORRECT, COMPASSX, SIOPACK
	OCC 's
	TRA
	OCC 's
	TRA
7 9	SNAP, xxx
7 9	TRACE, xxx
7 9	RUN, 2, 100, 7
	IDENT
	:
	:
	:

corrections for COMPASSX

corrections for SIOPACK

1	10
7 9	ALDAP (D),L,X
7 9	SNAP,xxx
7 9	TRACE,xxx
7 9	RUN,2,50,7
	ALGOL programs
	SCOPE

## 2.6 LIBRARY STATEMENT

With the library statement, the programmer may define an auxiliary library tape during a job.

```
7  
9 LIBRARY, u
```

u logical unit number 70, 71-79.

Logical unit numbers 71-79 define auxiliary library tapes, and 70 is used to return to the standard library tape during a job. However, the standard library tape remains available to SCOPE for I/O routines, debugging aids and the system itself while the auxiliary library feature is in use. All programmer requests for loading undefined entry points will call those routines from the auxiliary library tape.

When an auxiliary tape is defined with the LIBRARY statement, the table of contents for that tape is transferred to high core and is bounds protected. When end of job or the next LIBRARY,70 statement occurs, the tape and core are released, but the tape is not unloaded so that it may be referenced by the next job if necessary.

To assure that the proper tape has been mounted, an EQUIP statement should be used defining the label of the requested auxiliary tape. If there is no EQUIP statement, any library tape may be used as the auxiliary library, but only one auxiliary library tape is permitted during each run. A LIBRARY,71 statement followed in the same job by a LIBRARY,72 statement releases logical unit 71 and logical unit 72 becomes the auxiliary library tape.

The programmer determines the logical unit number (70-79) of the library unit currently in effect by specifying U = 0 in a library request (Section 3.6.).

2.7  
**AET  
STATEMENT**

If conditions arise which alter the availability of the physical equipment, the Available Equipment Table (AET) entries in storage may be changed by the AET control statement. When changes are made by the AET statement, only the table in storage is altered; the original table recorded on LIB is not affected.

The operator or programmer may use the AET control statement to: (1) obtain a listing of the contents of the Available Equipment Table, (2) alter the status of a unit, and (3) establish or alter an entire table entry. In the following formats, e is the octal ordinal of the entry in the table.

To obtain a listing:  $\begin{matrix} 7 \\ 9 \end{matrix}$ AET, e, m

If e is blank, the entire table is referenced.

m is the unit on which the entry or table is to be written:

m = blank     the output comment unit

m = OUT       the listable output unit

To alter the status of a unit:  $\begin{matrix} 7 \\ 9 \end{matrix}$ AET, e, a

a is the availability of the unit:

a = DOWN     the unit is unavailable

a = UP        the unit is available

The operator uses DOWN to indicate the equipment is not operating. When the condition is corrected, the UP entry is made.

To alter an entire entry:  $\begin{matrix} 7 \\ 9 \end{matrix}$ AET, e, oooooooooooooooooo

o's represent 16-octal digits which is the value replacing the previous value for entry e. See Appendix A for the format of the Available Equipment Table.

After changing a table entry, SCOPE writes the new entry on OCM (Appendix C).

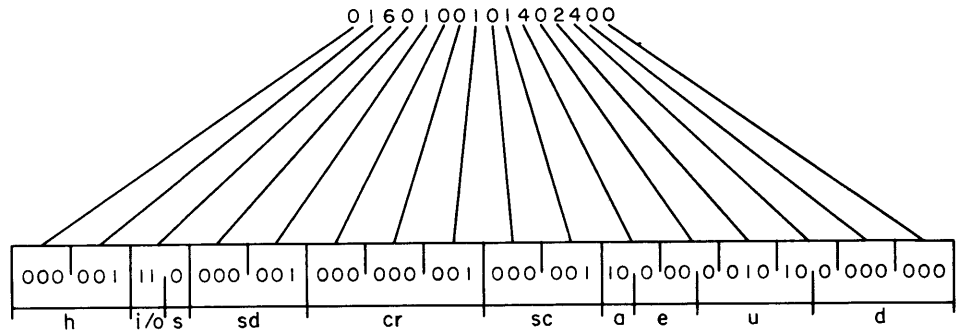
**Examples:**

$\begin{matrix} 7 \\ 9 \end{matrix}$ AET, 5            The 5th entry of the table is printed on OCM.

$\begin{matrix} 7 \\ 9 \end{matrix}$ AET, ,OUT        SCOPE writes the entire table on OUT.

$\begin{matrix} 7 \\ 9 \end{matrix}$ AET, 15, DOWN    Operator notifies SCOPE that the 15th unit is down.

$\begin{matrix} 7 \\ 9 \end{matrix}$ AET, 15, UP       Unit represented in the 15th entry is again available.



- |     |  |    |                                    |
|-----|--|----|------------------------------------|
| h   | magnetic tape                                      | cr | identifies controller ordinal      |
| i/o | both input and output tape<br>may be used for      | sc | Satellite control channel<br>field |
| s   | unit is accessible to SCOPE                        | a  | unit is assigned to SCOPE          |
| x   | unit is not capable of asyn-<br>chronous interrupt | e  | Satellite control channel<br>field |
| sd  | identifies driver ordinal                          | u  | unit AET ordinal                   |
|     |  | d  | driver ordinal                     |

If unit is to be used for input only, i/o = 10 rather than 11. The statement:  
<sup>7</sup>AET,12,0140100101402400 would make the necessary change in AET.

## 2.8 END REEL STATEMENT

<sup>7</sup>END REEL

This statement terminates the current reel of the standard input unit and may appear between runs on the INP. It causes SCOPE to locate the next INP reel or, if in Satellite mode, to accept the standard input unit from the Satellite. An end-of-tape also causes SCOPE to locate the next INP reel, and does not require the END REEL statement. It does not terminate a job; no job accounting occurs. END REEL should be used with discretion, usually the installation provides the END REEL statement.

An end-of-file mark must precede an END REEL statement.

In Satellite mode, END REEL cards are recognized but not processed by the Satellite computer.

**Examples:**

```
7  
9COMPASS, L, X  
.  
.  
.  
END  
SCOPE
```

----- physical end-of-tape mark

SCOPE will look for the next reel of the standard input unit and continue processing the COMPASS program.

```
7  
9COMPASS, L, X  
.  
.  
.  
END  
SCOPE
```

<end-of-file>

```
7  
9ENDREEL
```

SCOPE will look for the next reel of the standard input unit. If there are more COMPASS programs to be assembled in this job, the first control statement on the next reel of the standard input may be:

```
7  
9COMPASS, X, L
```

2.9

**ENDSCOPE  
STATEMENT**

```
7  
9ENDSCOPE
```

This statement signals SCOPE that the job stack is completed, and job accounting for the last job is recorded. ENDSCOPE appears on the standard input unit following the series of jobs to be processed. If INP consists of more than one reel, a physical end-of-tape causes SCOPE to transfer from a current reel to the next and ENDSCOPE appears on the last reel. The operator may enter this statement on the input comment unit to terminate processing prematurely.

SCOPE releases INP, OUT, PUN, and ACC when ENDSCOPE is encountered. An end-of-file mark must precede an ENDSCOPE statement.

2.10  
LOADING  
OBJECT  
PROGRAMS

A description of the loader is in chapter 5. The `LOADER` request (3.4) allows the programmer additional control over loading operations.

STANDARD  
INPUT

A binary object program on the standard input unit is loaded into storage unless a preceding control statement specifies other processing. All binary program cards up to a `SCOPE` control statement or two transfer cards are loaded.

```
7  
9JOB,1234-A, DDS, 7
```

binary object program

.  
. .  
. .

OTHER UNITS

Relocatable binary subprograms can be loaded into storage from programmer units or the load-and-go unit by the `LOAD` statement.

```
7  
9LOAD,u
```

u a logical unit number 1-49, or 69. When u is omitted, the standard load-and-go unit (69) is implied.

When the `LOAD` statement is encountered, `SCOPE` backspaces unit u one file and loads subprograms until an end-of-file, two transfer cards, or another control card is encountered. If the unit cannot be backspaced, `SCOPE` immediately loads the subprograms. `SCOPE` interprets a second transfer card as a loader terminator, but it is not required. If binary subprograms, transferred from the standard input unit and produced by compilation or assembly, are stored on the same logical unit during the job, only one end-of-file mark will be present and it will follow the last subprogram stored on the unit.

Binary subprograms may follow the `LOAD` statement if the loaded information was terminated by an end-of-file mark and not two consecutive transfer cards.



**Examples:**

<sup>7</sup><sub>9</sub>JOB, ACC77, AWS, 12

<sup>7</sup><sub>9</sub>LOAD, 36

<sup>7</sup><sub>9</sub>LOAD, 37

.  
. .

Binary subprograms are loaded from logical units 36 and 37. If there are two consecutive TRA cards, they must be the last records on unit 37.

<sup>7</sup><sub>9</sub>JOB, ACC77, ABC, 5

<sup>7</sup><sub>9</sub>LOAD

binary object subprogram

.  
. .

2.11  
**EXECUTING  
OBJECT  
PROGRAMS**

The RUN statement initiates program execution by transferring control to the object program in storage. This statement is required to execute all object programs. Library programs require a RUN statement only if debugging parameters are specified.

<sup>7</sup><sub>9</sub>RUN,t,p,r,m, TD

- t the execution time limit in minutes (maximum 2184 minutes). The entire job is terminated if the limit is exceeded. If t is blank, a constant time limit, determined by the installation, is supplied. If the run time limit is greater than the remaining job time limit, execution continues only until the job time is depleted. The run limit may not equal zero.
  
- p the maximum number of print or write operations which may be requested on the standard output unit during the execution. This includes debugging dumps and any other output during execution. The entire job is terminated if the print limit is exceeded. If the print line limit is blank, a constant print limit, determined by each installation, is supplied. The print limit may not equal zero.

r recovery indicator specifies area to be dumped if program does not proceed to normal completion. Appendix D gives recovery dump format.

<u>r</u>	<u>dumped area, written on standard output unit</u>
0 or blank	console
1	program and console
2	labeled common and console
3	program and labeled common and console
4	numbered common and console
5	program and numbered common and console
6	labeled and numbered common and console
7	console and all locations in all blanks except those occupied by SCOPE

m the memory map indicator. If m is blank, storage allocations after loading will be listed on the standard output unit. No map is written if m is any other character. See Appendix C for format.

TD Terminal dump. If TD is non-blank, a terminal dump will be provided at normal termination according to the specified recovery key. The rest of the job will be skipped.

**Example:**

<sup>7</sup><sub>9</sub>RUN,28,3000

Execution time limit is 28 minutes, and 3000 is the maximum number of print requests. The console dump, if the program is terminated, and the memory map are written on the standard output unit.

**PROGRAM  
TERMINATION**

The last executable statement in a program should return control to the operating system. This is accomplished through an EXIT request (Section 3.6) or a jump to the named transfer. (An exit is accomplished by jumping to the transfer address specified by the entry point on an END card in a COMPASS subprogram.) Programs that terminate by returning control to the operating system, with no abnormal conditions existing, terminate normally. Programs terminated by any other method are terminated abnormally.

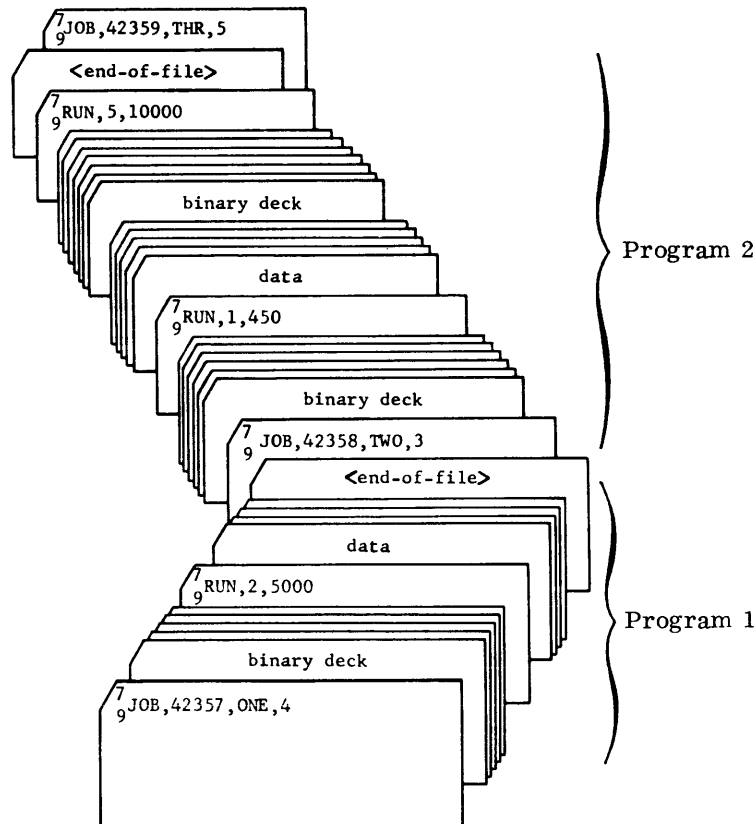
## END-OF-FILE CARD

End-of-file indicators are required to separate jobs and are frequently used to separate runs within a single job. If a job or run is terminated abnormally, SCOPE skips to the next end-of-file on INP and reads the following statement.

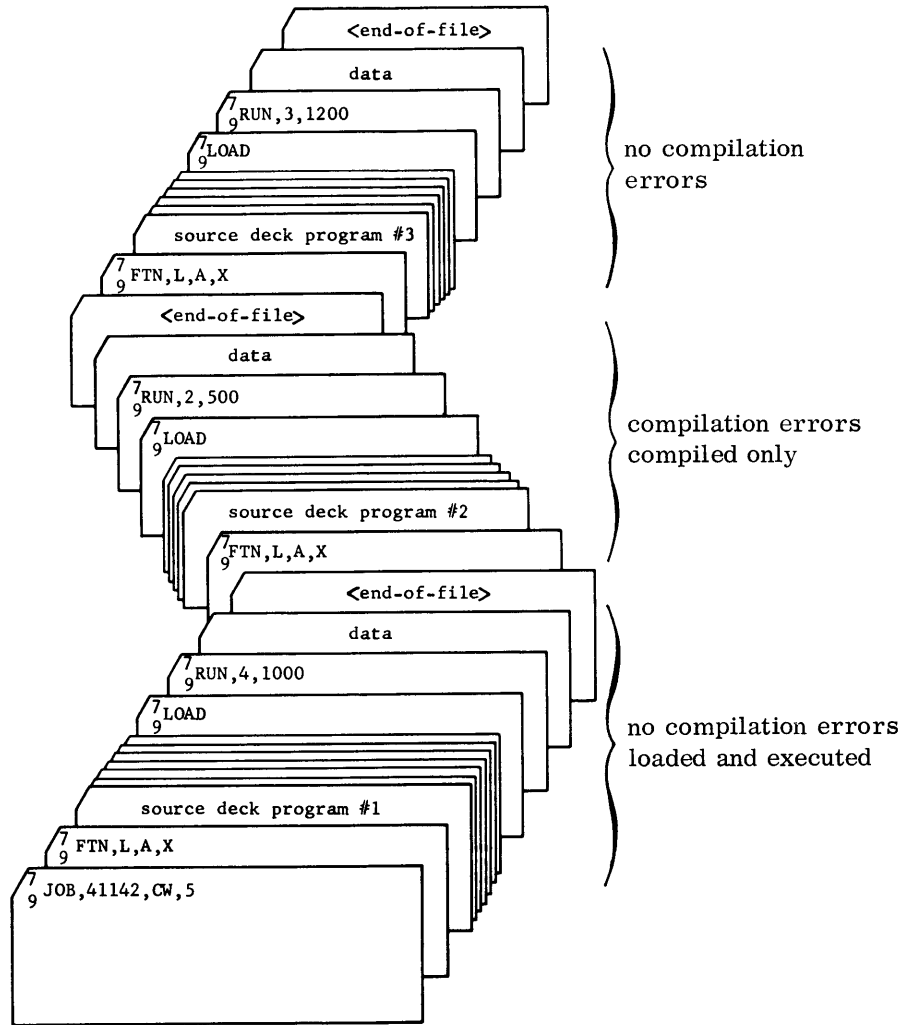
Runs should be separated by end-of-file indicators which must appear just prior to the time when control is returned to SCOPE from the running program. If program data follows the RUN card, the end-of-file comes after the data.

If errors occur during assembly or compilation of a program, loading of the job is not attempted. Subsequent assemblies or compilations of programs in the job are processed, however, if the entry point name control statement is preceded by an end-of-file.

When a program is running under Satellite mode or using the on-line card reader, an end-of-file is produced by a card with a 7,8 punch in column 1. Other peripheral processing programs may require a different punch configuration to produce an end-of-file mark on tape.



If program 1 terminates abnormally, SCOPE skips to the beginning of the next job. If program 1 runs to completion, but all of the data is not processed, SCOPE attempts to read the next data card, prints a diagnostic and skips to the next job.

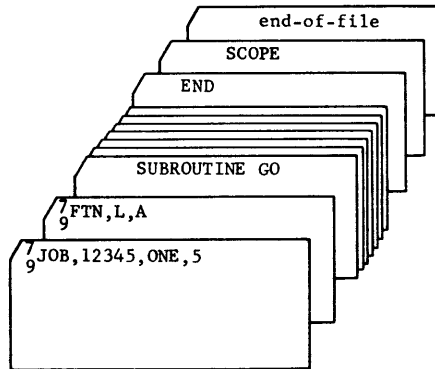


Since there are no errors in the first compilation, the object program is loaded and executed. Errors are encountered in the second compilation; after compilation is completed, SCOPE skips to the end-of-file mark and begins compiling the third program. After the third compilation, SCOPE skips to the next end-of-file mark. Had any of the runs terminated abnormally, SCOPE would have skipped to the beginning of the next job.

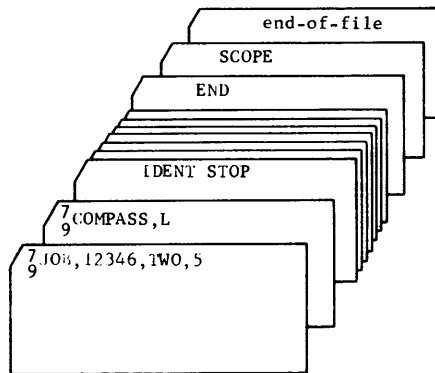
If subsequent compilations or assemblies are not to proceed when compiler errors occur, the runs should not be separated by end-of-file indicators.

2.12  
EXAMPLES OF  
DECK STRUCTURE

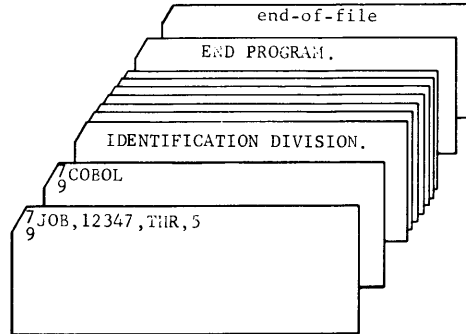
1) Compilation of a single FORTRAN subroutine.



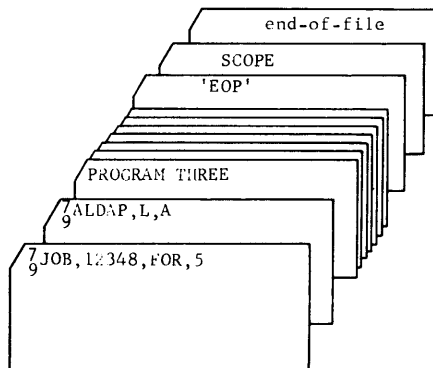
2) Compilation of a single COMPASS subroutine.



3) Compilation of a single COBOL program.

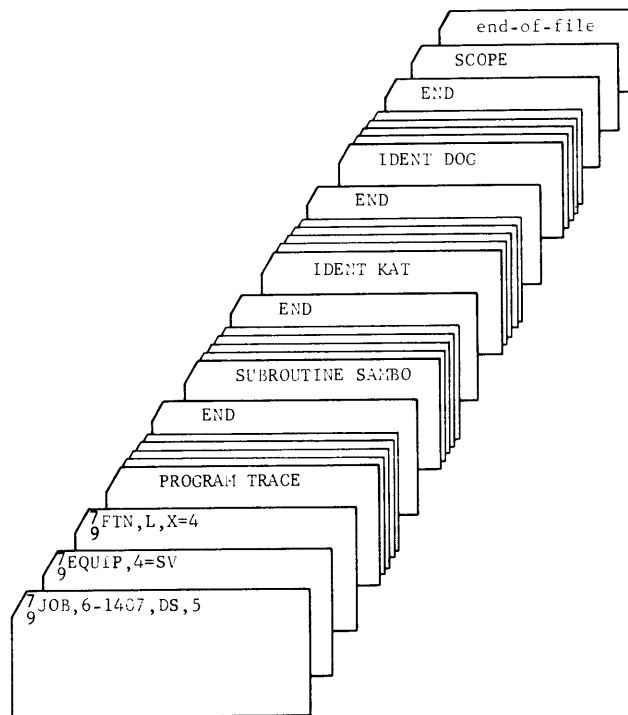


4) Compilation of a single ALGOL program.

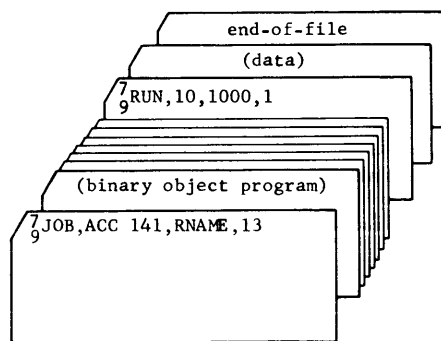


PROGRAM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	REMARKS			
ALGOL	P	R	O	G	R	A	M			n	a	m	e																			FIRST CARD OF SOURCE SUBPROGRAM(OPTIONAL)				
										'	E	O	P	'																			LAST CARD OF EACH SUBPROGRAM			
										S	C	O	P	E																			END OF COMPILE (ALDAP ONLY)			
COBOL										I	D	E	N	T	I	F	I	C	A	T	I	O	N	D	I	V	I	S	I	O	N	.	FIRST CARD OF SOURCE PROGRAM			
										E	N	D	P	R	O	G	R	A	M	.														END OF PROGRAM AND COMPILE		
COMPASS										I	D	E	N	T	n	a	m	e																FIRST CARD OF EACH SOURCE SUBPROGRAM		
										E	N	D																						LAST CARD OF EACH SOURCE SUBPROGRAM		
										S	C	O	P	E																				END OF ASSEMBLY		
FORTRAN										P	R	O	G	R	A	M	n	a	m	e															FIRST CARD OF EACH SOURCE SUBPROGRAM	
										S	U	B	R	O	U	T	I	N	E	n	a	m	e													
										F	U	N	C	T	I	O	N	n	a	m	e															
										E	N	D																								LAST CARD OF EACH SOURCE SUBPROGRAM
									S	C	O	P	E																						END OF COMPILE	

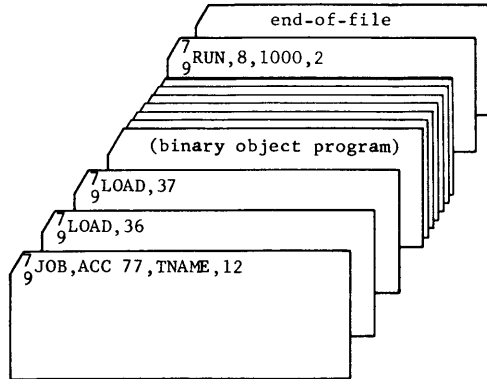
5) Compilation of a FORTRAN Program and several subprograms together with a COMPASS assembly. All binary object programs are placed on unit 4, which is saved.



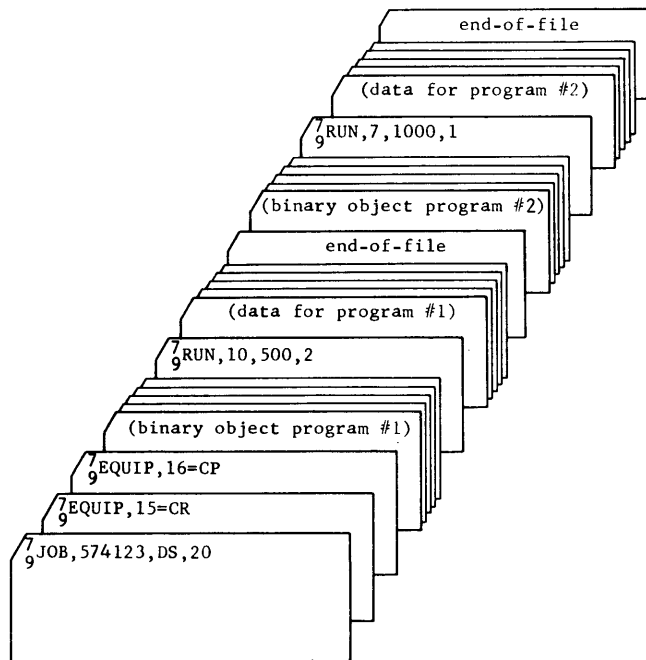
6) Execute directly from the standard input unit.



7) Load from two programmer units and INP.



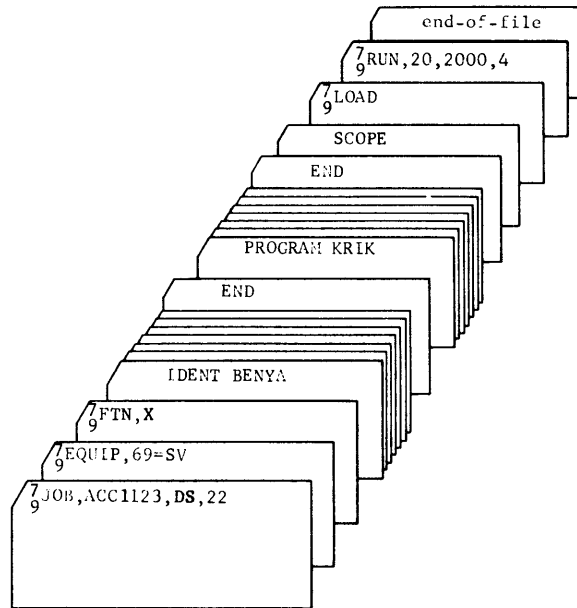
8) Sequential execution using EQUIP cards



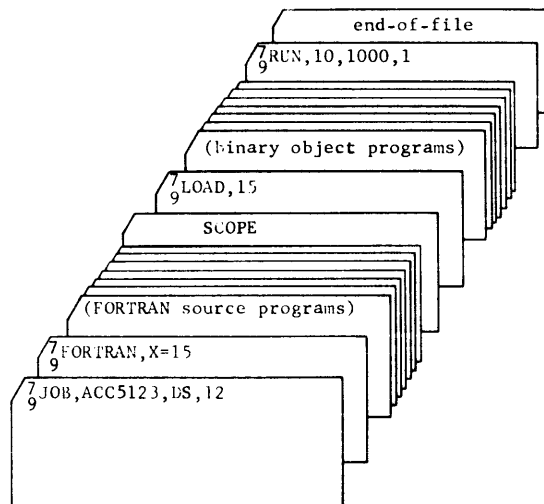
Changes made because EQUIP statements remain in effect for the duration of the job or until changed by another EQUIP statement.



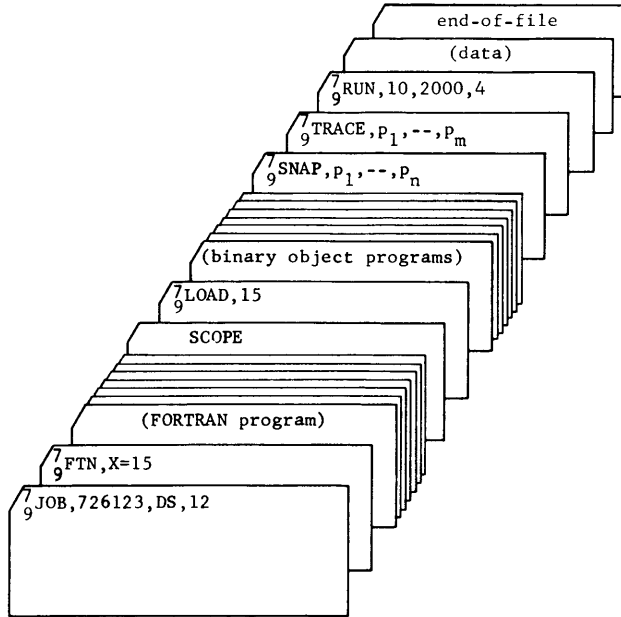
- 9) Assemble a COMPASS subprogram and FORTRAN subprogram on the load-and-go unit. Execute the program. Save the load-and-go tape.



- 10) Compile and load a FORTRAN program. Load a binary subprogram from INP. Execute.

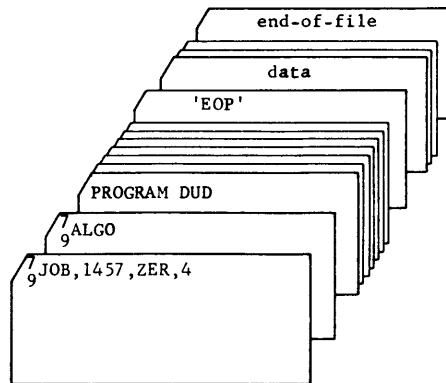


11) Include debugging aids.

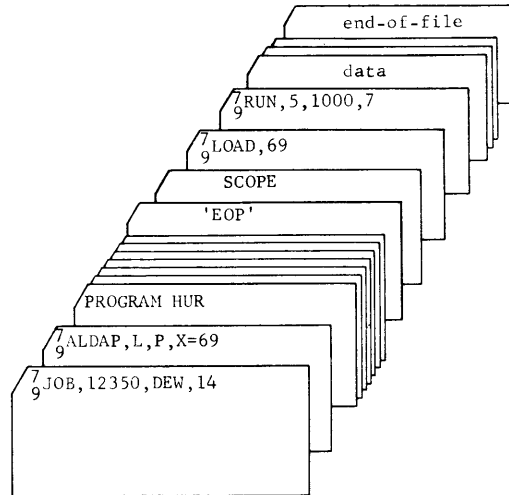


SNAP and TRACE cards precede the RUN card. (Refer to Chapter 4). The number of print requests includes SNAP or TRACE dumps.

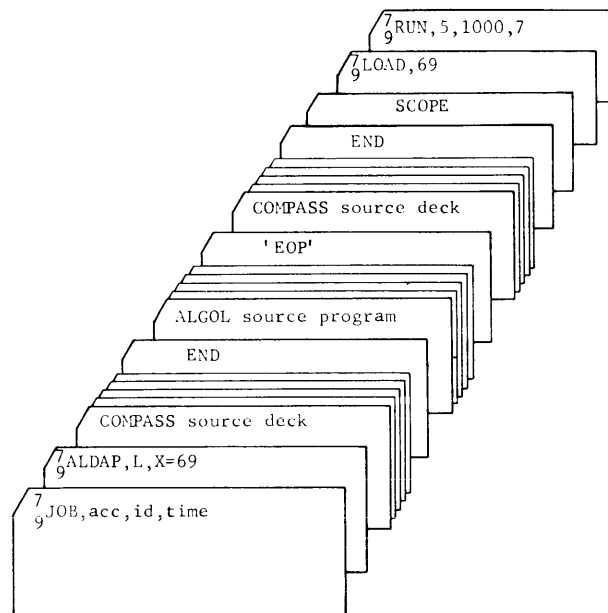
12) Compile and execute ALGOL program.



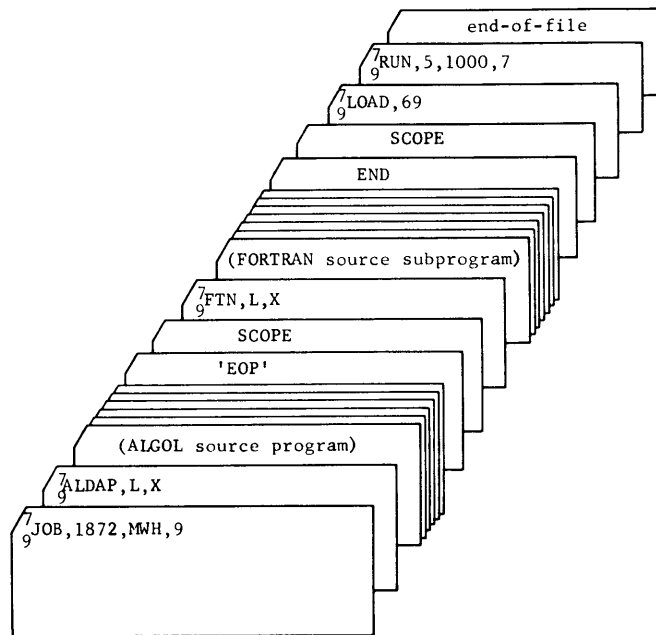
13) Compile and execute ALGOL program.



14) COMPASS subprograms in the form of subroutines may be assembled with an ALDAP compilation containing external declarations for the COMPASS subprograms. Neither a COMPASS nor FORTRAN subroutine may have a named transfer card, since the ALGOL program always has a named transfer card.



15) ALGOL programs may be compiled before or after FORTRAN programs for the same job. The basic operations for ALDAP, compile only, execute only and load-and-go are similar in format to those of other systems. A transfer address is generated by the ALDAP compiler and may not be provided by the programmer.



---

Programmer requests are statements which can be included only in assembly language (COMPASS) programs. They specify operations for input/output control, internal interrupt, clock interrupt, and special requests. They may be written as system macros or in any fashion which generates a calling sequence to SCOPE as outlined in Appendix B.

## 3.1 INPUT/OUTPUT REQUESTS

SCOPE processes all input/output requests, including read, write, equipment status checks, and tape handling, and performs the following operations:

- Assigns logical unit numbers to physical units
- Selects an available channel
- Stacks a request if a channel is not available
- Responds to external interrupts
- Initiates input/output operations
- Locates a continuation tape when needed to complete an input/output operation or initiates one when end-of-tape is reached

Parameters used in describing the requests are:

- u logical unit number, 1-79, or a mnemonic for a system or scratch unit. Mnemonics are given in section 1.4.
- cwa address of the I/O control word, or the first I/O control word in a chain. Control word format is shown in the 3600 Reference Manual.
- ra reject address to which control is transferred if the unit is unavailable. A reject address must always be specified. If an asterisk, \*, is given as the reject address, the request is repeated until the unit becomes available.
- ia address of the programmer's interrupt subroutine to which control transfers when an interrupt condition is sensed; this term may be omitted. If an abnormal condition occurs, the tape is stopped at the end of the control.

The logical unit number and the program addresses may be modified by the contents of an index register. The base operand (m) and the index register designator (b) are separated by a comma, and enclosed within parentheses (m,b). b may designate index registers, 1-6, or may specify indirect addressing, 7.

All addresses (control word, reject, interrupt) must be located within the same bank as the subprogram containing the READ/WRITE request. Any address may be defined as an external symbol; but the subprogram containing the associated entry point must be in the same bank (see BANK control statement).

Neither the reject nor control word address may be 0 or 77777<sub>8</sub>; the interrupt address may not be 77777<sub>8</sub>. If these addresses are used, the program will be terminated when they are detected by SCOPE.

The conditions causing request rejection are:

- Unit unavailable

- Request must be stacked, but stack already contains 25 requests

Input/output requests that SCOPE cannot handle are either rejected or cause the job to be abandoned. The following conditions cause job termination:

- Too many labels required for the job.

- System or equipment failure such as rejection by equipment supposed to be available or illegal entries in system tables.

- OCM not assigned or operable

- Illegal parameters in I/O calls such as:

  - logical unit not in the range 1-79

  - reject address of 0 or 77777

  - interrupt address of 77777

  - control word address of 0 or 77777

- Contradictory MODE request

- Output operation for a read only unit

- Request for an impossible function, (WRITE card reader)

- An operation in which the request specifies an illegal unit as indicated in the following table:

	Programmer Units 1-49	Scratch Units 50-59	INP 60	OUT 61	PUN 62	ICM 63	OCM 64	ACC 65	Satel- lites 66-68	LGO 69	LIB 70	Aux. LIB 71-79	SCR 80
READ			†	T	T		T		T				T
WRITE			T			T			T		T		T
REOT			T	T	T	T	T	T	T	T	T		T
WEOT			T	T	T	T	T	T	T	T	T		T
BSPF			T	T	T	T	T	T	T	T	T		T
BSPR			†						T				T
REWIND			T	T	T	T	T	T	T	T			T
SKIP				T	T	T	T	T	T	T	T		T
ERASE			T						T		T		T
MARKEF			T						T		T		T
UNLOAD		T	T	T	T	T	T	T	T	T	T		T
RELEASE			T	T	T	T	T	T	T		T		T
MODE									T				T
STATUS									T				T
LABEL		T	T	T	T	T	T	T	T	T	T		T
SAVE		T	T	T	T	T	T	T	T				T
RDLABEL		T	T	T	T	T	T	T	T	T	T	T	T
WRLABEL		T	T	T	T	T	T	T	T	T	T	T	T

T job terminated if request specifies this unit.

† READ or BSPR beyond EOF on 60 causes job termination.

## READ/WRITE

READ }  
WRITE } (u, cwa, ra, ia)

The programmer may direct the reading and writing of data with the READ/WRITE requests. If an end-of-tape condition exists, the READ/WRITE request will initiate a search for a new reel and release an old reel.

The direction of read may be designated by a parameter in the MODE request.

### **Examples:**

READ (INP, CONTROLA, SAM, INTRPT)

Data is to be read from the standard input unit; the control word is at CONTROLA. If the request is rejected, control is to be transferred to SAM. When the read operation is completed (or an abnormal condition occurs), control transfers to the interrupt subroutine at location INTRPT.

WRITE (OUT, CONTROLB, \*, INTRPTB)

A write operation is to be performed on the standard output unit. The control word is at CONTROLB, and the write request will be executed when OUT becomes available. When interrupt occurs, control transfers to INTRPTB.

## RDLABEL/ WRLABEL

RDLABEL }  
WRLABEL } (u, la, ra, ia)

When RDLABEL and WRLABEL are encountered, the tape is assigned and the label is read or written. If the tape is not at load point the job is abandoned. For RDLABEL, the label is transferred to the label area in core storage. The programmer may interrogate any of the ten words in the label area after a label has been read. For WRLABEL, SCOPE transfers the first four words of the label (provided by EQUIP or LABEL statement or SCOPE supplied) into the first four words of the label area (la to la+3). The last six words in the label area (la+4 to la+9) are written into the label in character positions 33-80. (See Sec. 1.8.) If no tape is found for a RDLABEL request and the operator types in NONE, the first word of the label area is zeroed and the normal return is taken.



## REOT/WEOT

REOT }  
WEOT } (u, cwa, ra, ia)

REOT and WEOT are tape movement controls which allow the programmer to read and write after the physical end-of-tape, before a continuation reel is assigned. If the request occurs before a physical end-of-tape, the reading or writing occurs and a logical end-of-tape condition is set. At the next READ or WRITE request, a continuation reel is established.

If a REOT or WEOT request falls between a LABEL and any other request, a new unit is assigned and the operation is performed.

### **Example:**

REOT (25, RDCWA, \*)

Logical unit 25 is read. After reading is completed, a logical end-of-tape condition is set. Upon the next READ request for unit 25, a continuation reel will be assigned.

## TAPE CONTROL REQUESTS

control name (u, ra, ia)

Requests may be stacked.

Control names applicable to magnetic tape units are listed below:

BSPF	Backspace one file.	} If the unit is unassigned these requests are bypassed.
BSPR	Backspace one record.	

BSPR and BSPF clear a logical end-of-tape condition. The physical end-of-tape may remain set if the unit is not back-spaced beyond the end-of-tape mark.

REWIND Rewind to load point. REWIND moves the currently assigned physical unit to the load point.

SKIP Skip to end-of-file or end-of-tape.

ERASE Erase 6 inches of tape.

MARKEF Mark end-of-file.

### **Examples:**

REWIND (20, RETURN, INT)

MARKEF (S2, REJECT1, INT2)

## UNLOAD

An UNLOAD request may be used to rewind and unload a magnetic tape unit.

UNLOAD (u, ra, ia, c)

If the logical unit represents a multi-reel assignment, only the physical unit presently assigned will be affected by the UNLOAD request.

The release code, c, specifies the disposition of the unit assignment after it has been unloaded:

0            unit assignment released  
non-zero    unit assignment not released

If an interrupt address is specified, a non-zero release code is assumed.

## RELEASE

This request releases the assignment of a logical unit and directs the disposition of the current physical unit.

RELEASE (u, ra, c)

The release code, c, specifies the disposition of the physical unit currently assigned to the logical unit:

0            dispose of physical unit according to previous declaration; if none is given, the tape is rewind.  
non-zero    rewind the physical unit and release the assignment, but do not dispose of the tape.

## MODE

A MODE request defines the usage of a tape unit or specifies density or recording mode for the unit. A MODE request can be honored only if the unit is available; if it is unavailable, control returns to the reject address.

MODE (u, ra, s, f, d, dr)

Usage, s, specifies an operating condition for the unit:

RW        (read and write) all legal requests will be performed.  
BY        (bypass) all requests except STATUS or MODE will be treated as no operation until the end of the job.  
RO        (read only) WRITE, WRLABEL, WEOT, MARKEF, or ERASE requests will be rejected.

Format, f, is specified by BCD for even tape parity or BIN for odd tape parity.

Density, d, is specified:

- HY hyper density tape (800 bpi) or highest possible density
- HI high density tape (556 bpi)
- LO low density tape (200 bpi)
- OP operator: Use density of mounted tape

Direction, dr, of tape; if not specified, normal is assumed.

- RV reverse direction for the READ or BSPR requests. Data is stored according to the control word address with no alteration.
- ND normal direction for READ or BSPR requests.

One of each of the designators, usage, format, density, or direction may be specified in a single MODE request; all four are not required. A blank, 0, designator terminates the list. If more than one designator for s, f, d or dr is indicated in a single mode request, the job is terminated.

**Examples:**

MODE (24, INSTR, RW, BIN, HY)

Logical unit 24 will perform all legal requests, with odd tape parity and hyper density. Reject address is INSTR.

MODE (25, REJECT, BY)

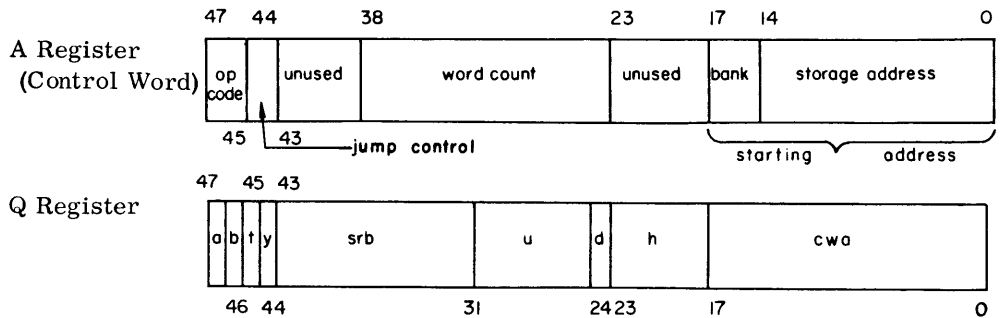
All requests except STATUS and MODE will be treated as no operation.

**STATUS**

STATUS (u, M)

The programmer may request the status of a logical unit at any time during operation. M designates the master unit (Equivalence Declarations 2.4)

The STATUS reply is entered in the A and Q registers as follows: (The same information is contained in A and Q upon entering user's interrupt subroutines for input/output operations.)



a	physical unit availability indicator	0, unit may accept request 1, unit in operation, or request is stacked
b	physical unit busy indicator	0, not busy 1, busy
t	magnetic tape indicator	0, unit is magnetic tape 1, unit is not magnetic tape
y	bypass indicator	0, unit is not bypassed 1, unit is bypassed

srb are the status reply bits given at the last reference to the unit; srb depend upon hardware type.

u is a logical unit number assigned to this physical unit, or the number of the logical unit specified in the last I/O request for this physical unit.

Since a physical unit may be referenced alternately by several logical units, the logical unit appearing in the reply may not be identical to the one given in the STATUS request. If the physical unit is available, u will be the logical unit number of the request; if not, u is one of the equivalent logical units.

If master status is requested, u will contain the master logical unit number.

d and h depend on the value of t

if t = 0, d and h give reel number (1-99) of the magnetic tape

if t = 1, d is the driver indicator

d = 0, no driver for unit

d = 1, driver required for unit; h is the hardware type of physical unit.

cwa is the current or last content of the control word address register of the data channel governing the unit (displayed in A register). If the request is given during processing, the control word is taken from the communication module. It reflects the latest word count and storage address.

The ab indicators, when combined, have the following meaning:

ab = 00 unit may accept request

ab = 01 unit is available, but interrupt for previous I/O request has not been processed

ab = 10 an I/O request is stacked

ab = 11 unit is in operation

STATUS REPLY BITS

status reply bits (octal)	362X Magnetic Tape Controller	3655 Printer Controller	3659 Printer Controller	3641 Card Reader Controller	3649 Card Reader Controller	3644 Punch Controller	3682 Satellite Coupler	731 Console Typewriter
xxx1	ready	ready	ready	ready	ready	ready	flag 0	ready
xxx2	read/write control (and/or) unit busy		busy	busy	busy	busy	flag 1	busy
xxx4	write enable			binary card	binary card		flag 2	upper/lower case
xx1x	end-of-file		paper out	end-of-file	end-of-file card		flag 3	
xx2x	load point		last line on form	feed failure	stacker full or jam, or fail to feed		flag 4	
xx4x	end-of-tape†			stacker full	hopper empty		flag 5	end-of-line
x0xx	200 bpi density 0 in bit 6							
x1xx	556 bpi density 1 in bit 6			hopper empty	end-of-file switch	fail to feed	flag 6	
x2xx	800 bpi density 1 in bit 7		ready and busy	amplifier failure	ready not busy	ready and not busy	flag 7	
x4xx	lost data		end of operation	end of operation	end of operation	O. D. Other Division	computer running	
1xxx	end of operation		abnormal end of operation		abnormal end of operation	abnormal end of operation	O. D. read	
2xxx	vertical parity error				compare or pre-read error	compare error	O. D. write	parity error
4xxx	reserve reject	reserved for other control	reserved reject		reserve reject	reserve reject	O. D. parity error	

†The end-of-tape bit will be set when the physical end-of-tape has been sensed or when the logical end-of-tape has been defined in the program. Logical end-of-tape may be defined before or after the physical end-of-tape has been sensed. The logical end-of-tape is set if a RFOT, WFOT, or LABEL request for a unit is given before physical end-of-tape.

If the unit is in operation (ab = 11), the reply describes the dynamic condition of the unit. For all other values of ab, the reply reflects the condition of the unit at the end of the last I/O operation.

If the unit is in operation (ab = 11) during an interrupt subroutine, the user should not wait for ab to change by repeating the STATUS request; since an interrupt on that unit cannot be processed (which may change ab) until control is returned to SCOPE. Any request but STATUS may be given to allow SCOPE to process interrupts.

When a logical unit has been bypassed, y is set to 1 and u is the logical unit to which the physical unit has most recently been assigned; the rest of the reply is zero.

If STATUS is given on an unassigned logical unit, A and Q are zero, except for the u field which contains either the called or the master logical unit.

**Example:**

The logical units are equated by EQUIP statements such that 2 is equivalent to 6, and 6 is the master unit,

STATUS (2) gives the status of logical unit 2, with 2 in the u field of the Q register.

STATUS (2, M) gives the status of logical unit 2, with 6 in the u field of the Q register. The status is identical in both examples, except for the u field.

**LABEL**

This statement provides identifying information for tape labels.

LABEL (u, addr, edition, reel, code)

logical unit            decimal number

addr                    address of the first of two computer words containing the name. Name may be 14 characters, alphabetic, numeric, or spaces; or it may be \*nn, where nn is a logical unit number less than 80. If the first two characters are \*\*, the tape is declared unlabeled.

edition number        1-99. If not specified, blanks are written in the output label; or any edition is accepted on an input label.

reel number            1-99. If not specified, reel 1 is written on an output label; or the lowest numbered reel is read from an input label.

code                    000-999. For an output tape, this retention code is the number of days tape is to be saved. 999 indicates infinite retention. If unspecified, zero is assumed which indicates immediate availability.

                          mmdyy    For an input tape, this code is the date the tape was written.

If a tape is not named, the logical unit number will be placed in the label. This number is either the master logical unit or \*nn specified in place of the name in the LABEL request. For \*nn, nn may not be used as the logical unit number in programmer requests to refer to the tape.

When a LABEL request is given, the assignment of the current reel is released, and the reel is rewound. At the end of the job, the reel is disposed of according to the disposition declaration.

Normally, a single LABEL request suffices for a unit throughout a run; it must precede the first I/O operation on the unit. When the first READ or RDLABEL request is given, the specified input label must match the label on the input tape. SCOPE writes the complete label at the first output request. Label information may be given in an EQUIP statement rather than in a LABEL request.

**Example:**

LABEL (15, =H\*10^~~~~)

Logical unit 15 will contain a label with a logical unit number of 10. The five spaces are required since =H defines a Hollerith literal of 8 characters. Logical unit 15 must be used in subsequent programming requests.

Since edition number and reel number are unspecified, the edition number will be blank and reel will be number 1 for output, or the lowest reel for input. Retention code is zero.

**SAVE**

The programmer may specify that a tape be saved at the completion of the job.

SAVE (u, U)

The SAVE request may be given at any point in the program as it does not inhibit reading and writing on the unit. At the end of the job, the current reel of the saved logical unit is unloaded and a message directs the operator to reserve the tape for the programmer. For a multi-reel, saved logical unit, each reel is unloaded and saved as it is completed or released. Save may also be requested in an EQUIP statement.

U changes this request to UNSAVE.

**Examples:**

SAVE (29)

SAVE (LGO)

SAVE (LGO, U)

UNSAVE (u)

This request which cancels any previous SAVE or EQUIP SV request may be given at any point in the program. The tape is released at the end of job for subsequent use. Unexpired retention codes can be overridden only by an UNSAVE request after the tape is assigned.

## 3.2

**STACKING OF REQUESTS**

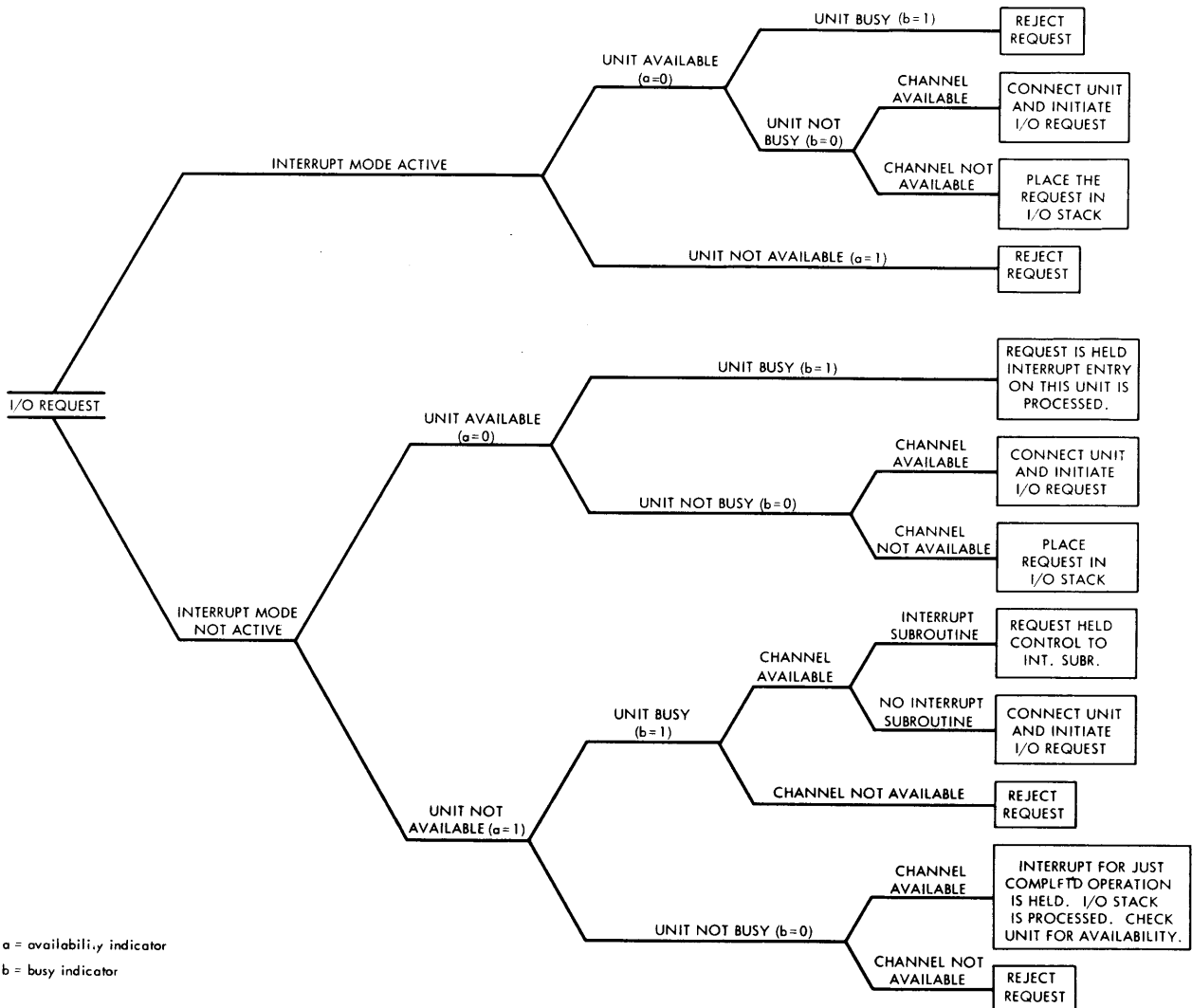
When an I/O request is given (see diagram) SCOPE determines whether it occurred while the interrupt mode was active or whether it came from an interrupt subroutine (interrupt mode not active).

If a request is made while the interrupt mode is active, and SCOPE finds that the unit is not available, the request is rejected. If the unit is available and not in operation, SCOPE will determine if a channel is available to which the unit may be connected. If so, the unit is connected and the request initiated. If no channel is available, the request is placed in the I/O request stack. In either case, a normal return is made to the calling routine.

When the I/O request comes from within an interrupt processing routine (interrupt mode not active), SCOPE determines availability of the unit. If the unit is available, but busy (an interrupt on this unit is being held), the request will be held while the interrupt is processed. If the unit is available and not busy, SCOPE determines if a channel is available to connect to the unit. If so, it is connected and the request is initiated. If no channel is available, the request is added to the I/O stack.

When SCOPE determines that the unit is not available and it is not busy (an I/O request on this unit is already in the stack) but no channel is available, the request is rejected. If a channel is available, the interrupt for the just completed operation is held and the stacked requests are processed. The I/O request is again routed through unit and channel availability checks. If an operation is being performed on the requested unit (unit not available and busy) SCOPE checks channel availability to see if the operation has been completed. If the channel is not available, the request will be rejected. If the channel is now available (operation has been completed) and an interrupt subroutine was not specified, the channel will be connected and the request initiated. If an interrupt subroutine was specified, the current request is held while the interrupt is processed.





### HELD REQUESTS AND INTERRUPTS

After an interrupt processing subroutine has been completed, SCOPE determines if any I/O requests or interrupts are being held. All held requests are processed (on a last-in-first-out basis) before each held interrupt. This is done so that the requests held during the execution of an interrupt subroutine may be honored before the next interrupt routine begins processing. When all of the held requests and held interrupts have been processed, SCOPE returns control to the main program.

### 3.3

#### EXTERNAL INTERRUPT CONTROL

If an interrupt address is specified, control will be transferred to that address at the end of the operation or upon abnormal condition interrupt. Before giving control to the interrupt address, SCOPE stores the A and Q registers and enters the control word in the A register and the unit status in the Q register (see STATUS 3.1). The Q register and the index registers are set to their values at the time of the interrupt. Control will be transferred to the interrupt address by a bank return jump. The programmer returns control from the interrupt processing routine to SCOPE by returning to the interrupt address. Upon regaining control, SCOPE processes any other interrupts, restores the A and Q registers, and returns to the running program. The Q register and the index registers must be preserved by the interrupt subroutine.

Input/output operations may be requested from within interrupt subroutines. However, if a request is given in an interrupt subroutine, the end of operation interrupt for that request will not be processed until the interrupt subroutine has been completely processed.

### 3.4

#### INTERNAL INTERRUPT CONTROL

Four controls are available for handling internal interrupt features. SELECT indicates the type of interrupt and the location of a routine to be entered when that interrupt occurs. REMOVE releases the interrupt. An interrupt address may also be reassigned, by an indirect technique, to a previously selected location. BOUND sets storage area limits outside of which references are not to be made and the address to which control is transferred if the bounds are violated.

UNBOUND releases the last set of bounds and reimposes the bounds that were in effect before the last BOUND request.

Program address parameters in all internal interrupt requests may be modified by the contents of an index register by enclosing within parentheses the program address and the index register designator separated by a comma, (m,b). The program address, m, may be any legal COMPASS address field expression. If b is 1-6, the address will be  $m + (b)$ . If b is 7, indirect addressing will be used.

All program address parameters must be located within the same bank as the subprogram containing the internal interrupt request.

**SELECT/REMOVE** SELECT designates the specific interrupt; when the interrupt occurs, a jump is made to the interrupt address. If a previous SELECT request with the same interrupt had been made, its interrupt address will be saved in the current SELECT calling sequence. When the interrupt occurs, further interrupts are locked out and a bank return jump is made to the interrupt address. Control is transferred from the interrupt subroutine to the monitor by a jump to the interrupt address.

SELECT (interrupt, address)

Address the location to which control is transferred when the interrupt is detected.

Interrupt

SHIFT	shift fault
DIVIDE	divide fault
EXOV	exponent overflow fault
EXUN	exponent underflow fault
OVER	fixed point overflow fault
ADDR	*non-existent address fault
M1604	1604 mode alert
TRACE	trace mode alert
INST	*illegal instruction fault
OPER	*operand parity fault
MANUAL	manual interrupt alert
ABNORM	abnormal termination

REMOVE (interrupt)

REMOVE removes the specified interrupt and saves the interrupt address declared in the SELECT request for that interrupt. The interrupt address is saved in the current REMOVE calling sequence.

If there was no SELECT, REMOVE acts as a NOP.

SELECT (I, SELECT or REMOVE address)

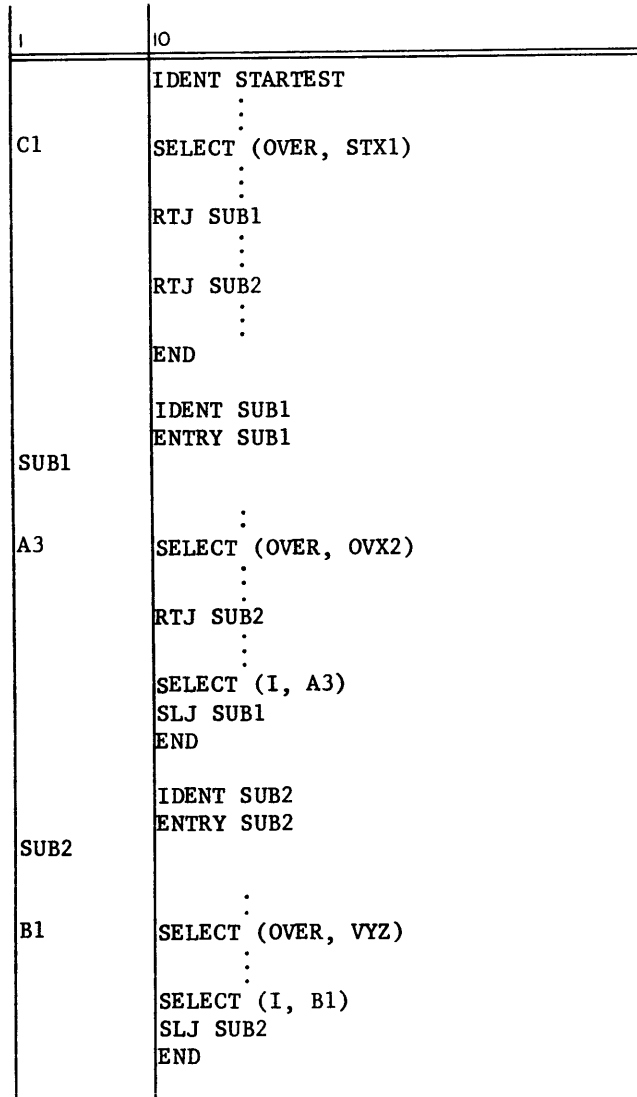
I specifies an indirect selection of the interrupt address assigned before the specified SELECT or REMOVE.

SELECT or REMOVE address is the location at which a previous SELECT or REMOVE request was made.

---

\*If these interrupts are not selected by the programmer, they will terminate the program.

SELECT indirect reselects an interrupt address designated by the SELECT or REMOVE instruction in effect prior to the instruction at the specified address. The indirectly selected interrupt address will be the address contained in the SELECT or REMOVE request, for the same interrupt, which was in effect prior to the request at the specified address.



Indirect interrupt select is requested in subprograms SUB1 and SUB2 before control is returned to the calling subprograms. Therefore, in each of these routines, a new interrupt address is taken when the routine is entered; and the previous interrupt address is reinstated before control returns to the calling program. The indirect interrupt request in SUB1, for example, removes the SELECT at address A3, leaving in effect the last interrupt address (STX1) specified for OVER before that SELECT was given. The indirect interrupt request in SUB2 removes the SELECT at B1 leaving in effect the last interrupt address, OVX2. Indirect SELECT allows the programmer to reinstate the interrupt address in the calling program before he transfers back to it and without knowing to which program the return will be made.

## ABNORM

### ABNORM

The ABNORM interrupt allows the programmer to specify an alternate interrupt subroutine to be entered before SCOPE terminates a job for an abnormal condition. If the job terminates abnormally and the ABNORM interrupt has been specified, control transfers to the interrupt subroutine; if the programmer does not specify ABNORM, SCOPE terminates the job immediately. Within the ABNORM interrupt subroutine, the programmer can take steps to provide a clearer indication of the job conditions. At the end of the interrupt subroutine, control returns to SCOPE by a jump to the interrupt address, and SCOPE provides the requested recovery dump.

When control transfers to the ABNORM interrupt subroutine, several conditions are in effect. The interrupt system is active; internal interrupts have been cleared. All previous I/O operations have been terminated; any held requests, stacked requests, held interrupts or interrupt subroutines have not been processed. The bounds register has been reset to the original programmer bounds. A fixed time limit and print limit, established by the installation, is imposed on the programmer's subroutine.

An abnormal condition, such as programmer area destroyed, or bad unit, may preclude successful execution of the ABNORM interrupt subroutine. If a second abnormal termination occurs during the ABNORM interrupt subroutine (for instance, time limit exhausted), the recovery information from the first termination is lost, information is given for the second termination, and the job is abandoned.

The ABNORM interrupt subroutine will not be entered following AUTOLOAD recovery.

## BOUND/ UNBOUND

SCOPE is protected by upper and lower bounds. The upper bound is the highest location in the highest numbered bank available, excluding the system Table of Contents and I/O drivers. The lower bound is the lowest numbered location in bank zero excluding SCOPE and the loader, unless numbered common is assigned to the region usually occupied by the loader.

The BOUND request sets bounds outside of which any instruction reference will cause an interrupt. The first BOUND request executed in a program may set any BOUND range allowed by SCOPE. Subsequent BOUND requests must set bounds which lie within the previously set bounds. If any requested bounds overlay those previously set, the request is rejected and control is transferred to the reject address. If the bounds are accepted, the previous bounds are stored. The number of nested bounds is limited to 5. If the programmer specifies lower and upper bounds as zero, SCOPE returns the last setting of the BOUNDS register in the A register.

BOUND (lower bound, upper bound, reject address, interrupt address)

lower bound

upper bound

are addresses to which the memory bounds are set; bank terms may be included in these addresses. If they are omitted, (\$) is assumed. The lower and upper bounds cannot be indexed because of bank terms.

(\$) name, indicates that the bank already associated with name will be used.

reject address

is the location to which control is transferred if the bounds list is full (5 requests) or if the requested bounds do not fall within those previously set.

interrupt

address

is the location of the programmer's interrupt subroutine to which control is transferred when an interrupt condition occurs.

## UNBOUND

The UNBOUND request removes the bounds set by the last executed BOUND request and re-establishes the bounds previously set. UNBOUND cannot be used to remove the bounds set by SCOPE.

### **Examples:**

BOUND ( ( (\*) LIMIT1), LIMIT2, RA, IA)

The lower bound will be LIMIT1 in the bank containing the BOUND request. The upper bound will be LIMIT2 in the bank in which LIMIT2 is located. (\$) LIMIT2 is assumed.

BOUND ( ( (0) LOWEST), ( (0) HIGHEST), RA, IA)

BOUND ( ( (0) LOWER), ( (0) UPPER), RA, IA)

UNBOUND

The first set of bounds are LOWEST and HIGHEST. The next set, LOWER and UPPER, lie within the first set. UNBOUND removes LOWER and UPPER and reinstates LOWEST and HIGHEST.

3.5

**CLOCK INTERRUPT** Four controls handle clock interrupts. LIMIT imposes a time restriction. FREE releases the last time limit, and re-establishes the previous time limit. TIME returns the time remaining until the next time interrupt in the A register, and the time of day in the Q register. DATE returns the calendar date in the A register.

**LIMIT** The LIMIT request sets a time limit after which control will be transferred to the interrupt address.

LIMIT (du, ra, ia)

- du the duration in seconds of the time limit; milliseconds may be appended by giving the parenthesized expression (seconds, milliseconds)
- ra the transfer location if the limit is not accepted.
- ia the location to which control is transferred when the limit is reached. The interrupt subroutine is entered by a bank return jump. The interrupt subroutine should return to the interrupt address.

The reject address and interrupt address may be modified by the contents of an index register. (See 3.1).

No more than five limits may be in effect at one time; each must fall within the time set by the last executed LIMIT request. If the new limit is larger than the previous, control will transfer to the reject address. When each new limit is accepted, the clock comparison register is reset and the previous limit is stored. Upon time interrupt, the limit which caused the interrupt is released before transferring control to the interrupt address.

**Example:**

LIMIT ( (1000,500), RA1, IA1)

**FREE**

**FREE**

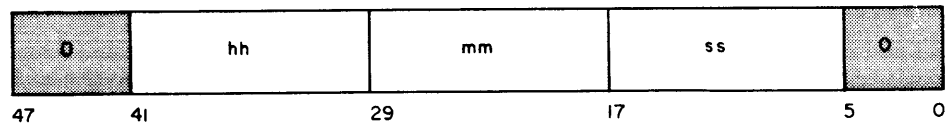
FREE releases the last time set by a LIMIT request (the smallest in the list of time limits) and re-establishes the next previous time set (the next smallest limit in the list). Limits set by SCOPE cannot be freed.

**TIME**

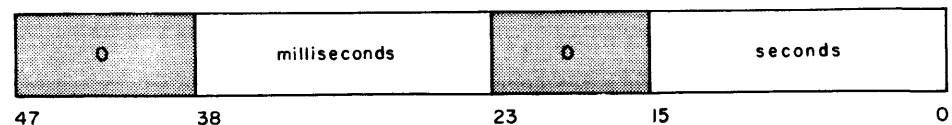
**TIME**

Upon receiving a TIME request, SCOPE enters the time of day into the Q register in BCD and the time remaining before the next time interrupt into the A register in binary.

The time of day is based upon a 24-hour clock (one minute before midnight is 235900) and is given in hours (hh), minutes (mm) and seconds (ss). This time is entered in the Q register in the format:



The time to the next interrupt is entered in the A register in the format:

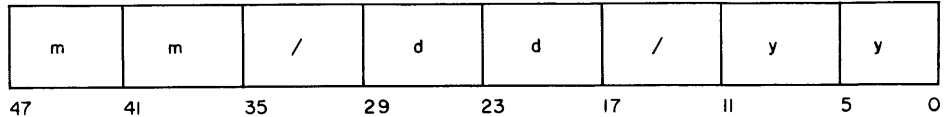




## DATE

## DATE

When a DATE request is received, SCOPE enters month, day and year into the A register, in BCD, in the format:



## 3.6

### SPECIAL REQUESTS

A running program may request SCOPE to position the library at a particular record or at the directory preceding it (LIBRARY). Requests are also available for calling the loader to load programs (LOADER), obtaining or setting memory limits (MEMORY), and for returning control to the monitor system (EXIT). An interrupt subroutine may modify the A and Q registers of the program at the time the interrupt occurred (HERESAQ).

## LIBRARY

LIBRARY may be used to position the library tape.

LIBRARY (u, ra, record name address, record number)

u	library logical unit number, 70 to 79 or 0.
ra	location to which control is transferred if the specified record is not found.
record name address	storage location of the first of four computer words containing the record name.
record number	signed integer, 0 through $2^{24}$ .

If u is zero, the other parameters are ignored; upon return to the program, the A register contains the logical unit number of the current library. Library unit, reject address, and record name address may be modified by the contents of an index register. (See 3.1)

The possibilities for the record number and location of record name in positioning the library are as follows:

<u>Record Name Address</u>	<u>Record Number</u>	<u>SCOPE Positions Library at:</u>
zero or blank	zero or blank	the next directory
non-zero	non-zero	record r of the series beginning with the named record
non-zero	zero or blank	the directory containing that record
zero or blank	non-zero	(see below)

LIBRARY may also specify the number of records which the user has moved the library tape. When the user has read or backspaced the library tape and intends to use either LIBRARY or LOADER requests later in the run, a LIBRARY request must be given indicating the current position of the library tape. The record name address will be blank or zero and the record number will be the signed number of records that the tape has been moved ( + forward, - backward).

**Example:**

LIBRARY (70, REJECTA, CTABLE, 1)

This request positions the library tape at the beginning of the record, CTABLE.

Assume that the user reads 2 records and then wants to find another table MATCHC. He must give:

LIBRARY (70, REJECTB, , +2)

LIBRARY (70, REJECTC, MATCHC, 1)

**LOADER**

**LOADER**

The loader request is used by a running program to call the loader. If the loader is not in storage, SCOPE will read it from the library into its customary position immediately following resident. Just prior to giving the request, the parameters (sec. 5.3) specifying what is to be loaded must be placed in the A and Q registers. SCOPE passes these parameters to the loader and retains control until that call to the loader is complete. Upon exit from the loader, the contents of A and Q are returned to the calling program for examination. The loader returns control to SCOPE through its entry point, and SCOPE returns control to the calling program.

If numbered common has been assigned to bank 0, the loader will overlay numbered common. SCOPE does not adjust the limits of available storage after loading the loader.

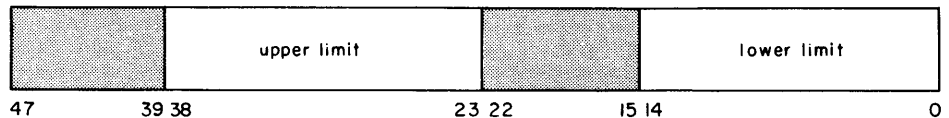
## MEMORY

The limits of available storage may be obtained or changed by the MEMORY request.

MEMORY (bank designator, lower limit, upper limit)

bank designator	0-7, *, or \$symbol. * designates the bank containing this request. \$symbol designates the bank in which symbol is located.
lower limit	any legal Compass Address expression in the range 1-77776 <sub>8</sub> .
upper limit	These must not be prefixed with bank designators or suffixed with index designators.

If either limit in the programmer request is zero, the other limit is not changed. Limits larger than SCOPE bounds can not be set. If the lower limit requested lies in resident, it will be reset to the first location following resident. If the requested upper limit is greater than the upper bound of SCOPE, the job is terminated. When the limits have been changed by MEMORY, the new limits are returned in the A register. If no limits are supplied in the request or if both limits are zero, the current storage limits for the specified bank will be entered in the A register in the following binary format:



A non-existent bank is indicated when the A register is equal to zero.

## EXIT

### EXIT

The EXIT request returns control from a running program to SCOPE. This causes a normal termination of the program as long as there are no abnormal conditions. Control may also be returned to the monitor system by transferring to the running program transfer address which is preset with an EXIT request by SCOPE.

## HERESAQ

## HERESAQ

The programmer's interrupt subroutine may modify the A and Q registers at time of interrupt with the HERESAQ request. When the request is given, new values specified by the programmer are stored temporarily. The A and Q registers are set to those values when the return is taken from the interrupt subroutine.

### Example:

```
      ⋮
      LDA      CON1
      LDQ      CON2
      HERESAQ
      ⋮
      UBJP     interrupt address
```

} programmer's  
interrupt  
subroutine

CON1 and CON2 are stored temporarily by SCOPE when HERESAQ is executed. UBJP transfers control to SCOPE. When SCOPE transfers back to the user's subprogram, CON1 and CON2 are the contents of A and Q.

## CORE

The limits of available storage in the bank in which the subprogram is located may be obtained or changed by the CORE request. This request differs from the MEMORY request in that no bank need be specified.

CORE (lower limit, upper limit)

Selected areas of storage may be dumped each time a particular instruction is encountered (SNAP) or before execution of each jump instruction in a designated area (TRACE). Recovery dumps may be designated for abnormal termination and a memory map, giving a listing of absolute addresses assigned to the program by the loader, may be obtained.

SNAP and TRACE dumps consist of a console scoop and a storage dump. (See Appendix C). The A and Q registers are printed in the mode requested. The index registers, bounds registers and P register are printed in octal. The interrupt register, interrupt mask register and switches are printed in binary.

Each printed line contains an absolute octal address, an octal address relative to the name in the first word address (fwa), and four to ten computer words, depending upon the mode requested. One or more lines of identical words are omitted.

## 4.1

### SNAP DUMP

The programmer specifies the addresses where snap dumps are executed, the frequency, and the areas to be dumped. SCOPE replaces the instructions at the dump addresses with jumps to the SNAP routine. The SNAP routine dumps the specified areas onto the standard output unit, executes the instructions originally at the dump address from within the range of the SNAP subroutine, and returns control to the program.

More than one snap dump may be specified for an address and any number of addresses may produce snaps. After the last snap is produced, normal program operation resumes. The only restrictions on the number of snap dumps are the print request limit in the RUN statement and the amount of available storage remaining after the program is loaded.

<sup>7</sup><sub>9</sub>SNAP,a,fwa,lwa,f,d<sub>1</sub>,d<sub>2</sub>,d<sub>3</sub>,id

- a the program address where the dump is initiated. The program address may be a program name or an entry point name plus or minus an octal displacement, p±n; p is an entry point or program name and n is an octal number. If a program name is used, the program address will be the first location in the program. If an entry

point name is used, the program address will be the entry point location. If the program and entry point names are identical, the program address will be the first location in the program.

The contents of that address must not be referenced or modified within the program. The contents of the cell must not contain an I/O control word, an RTJ or BRTJ jump which sets an address for reference, or an instruction which may skip exit.

fwa,lwa The first word address and the last word address of the area to be dumped may be:

- 1) A 6-digit absolute octal location; the left-most digit is the bank designator. If less than 6 digits are given, bank 0 is assumed.
- 2)  $p_y \pm n_y$ , a common block name, an entry point name, or a program name,  $p_y$ ; plus an octal displacement,  $n_y$ , relative to the name. If a common block name is used, it is enclosed in slashes: /name/. If the block name is blank, two slashes are given: //.
- 3)  $\pm n_x$ , an octal displacement relative to the previously declared entry point, common block, or program name on this SNAP card.
- 4) blank (fwa and lwa), no area will be snapped; the console will be snapped if C is suffixed to the mode designator. Where fields are omitted commas must be placed, unless no non-blank fields follow.

If fwa equals lwa, a console scoop will be given.

f the format of the dump on the standard output unit is designated by:

O or blank	octal dump
M	octal dump with mnemonic operation codes
I	fixed decimal dump, integer
S	floating decimal dump, single precision
D	floating decimal dump, double precision
B	BCD dump
C	is suffixed to the designator, if a snap of the console is to be included

$d_1, d_2, d_3$  control the start, stop, and frequency of the SNAP dump. A dump will be produced at the  $d_1$  encounter of address  $p \pm n$ , and at every  $d_3$  encounter thereafter until  $d_2$  is reached. If these parameters are blank, a dump is produced at every encounter of the address. If  $d_3$  is blank, a dump is produced at every encounter of the

address between  $d_1$  and  $d_2$ . The maximum value of  $d_1$ ,  $d_2$ , or  $d_3$  is 4096. After the last ( $d_2$ ) encounter, the original contents of the call are restored.

`id` is an optional identification for each dump on the standard output unit. It may be up to five alphanumeric characters.

The SNAP cards are placed immediately before the RUN card in the program deck.

### Examples:

```
7
9SNAP, ANNA, +5, +30, MC, 1, 100, 5, JACK
```

The area of storage occupied by locations ANNA+5 through ANNA+30 will be dumped. A dump is produced the first time location ANNA is encountered, and every 5th time thereafter until the 100th time. JACK is printed with each dump as identification. An octal dump with mnemonics and a console scoop are produced.

```
7
9SNAP, BETA, +0, +50
```

The snap is triggered by location BETA. The area of storage to be dumped is BETA through BETA+50. An octal dump will be produced every time BETA is encountered.

```
7
9SNAP, BETA, /SAM/, +50, S, 1, 20, 2, JM
```

The common storage area SAM through SAM+50 is dumped in floating decimal, single precision when BETA is encountered. A dump is produced the first encounter and every alternate encounter until the 20th. JM is the identification.

## 4.2

### TRACE DUMP

The TRACE statement produces a dump whenever jump instructions within a specified range of the program are executed. The dump will be written on the standard output unit in the same format as the snap dump.

```
7
9TRACE, a1, a2, fwa, lwa, f, d1, d2, d3, id
```

$a_1$  is the first address of the trace area; may be an entry point or program name plus or minus an octal displacement ( $p \pm n$ ).

$a_2$  is the last address of the trace area;  $a_2$  must be greater than  $a_1$ .  $a_2$  may be one of the following:

$\pm n_x$  an octal displacement relative to the program entry point or program name,  $p$ .

$p_y \pm n_y$  an octal displacement relative to program entry point or program name,  $p_y$ .

Any number of ranges ( $a_1$  to  $a_2$ ) may be specified. If two or more tracing limits overlap, the second supersedes the first. The contents of the address  $a_1$  may not be referenced or modified within the program nor may either  $a_1$  or  $a_2$  contain an input/output control word or a jump instruction (RTJ or BRTJ) which sets an address for return or any instruction which may skip exit. When the last trace is produced,  $a_1$  and  $a_2$  are restored and normal program operation resumes.

fwa is the first word address of the area to be dumped.

lwa is the last work address of the area to be dumped.

The parameters, fwa and lwa correspond to those of the SNAP statement.

f is the format of the dump on the standard output unit. The various designators are described following the SNAP statement.

$d_1$  specifies the number of times the area to be traced is passed through before a jump may produce the first dump.  $d_1$  must be less than 4096.

$d_2$  specifies the last time through the trace area that a jump instruction will cause a dump.  $d_2$  must be less than 4096.

$d_3$  specifies how often tracing occurs when passing through the trace area.  $d_3$  must be less than 4096.

The area is traced at the  $d_1$  encounter of  $a_1$ , and at every  $d_3$  encounter thereafter until  $d_2$  is reached. During tracing, the counter is not incremented until  $a_2$  is encountered; jumps to  $a_1$  in TRACE mode will not affect the count of the trace. If the parameters are blank, tracing is initiated at every encounter of  $a_1$ .

The specified area (fwa to lwa) is dumped before the jump instructions are executed. If the jump transfers control to a location outside of the tracing limits, trace output is halted. Upon returning within limits, trace output is resumed, beginning with the first jump instruction within the limits. If a jump instruction is located at  $a_1$ , it is traced; at  $a_2$ , it is not traced.

The TRACE cards are placed immediately before the RUN statement. If both SNAP and TRACE cards are used, their order is not significant, as long as they are the last cards before the RUN card.

**Example:**

```
7
9TRACE, ALPHA, BETA, +5, +30, MC, 1, 100, 5, JACK
```



Jump instructions in the range ALPHA - BETA will be traced. The locations BETA+5 through BETA+30 will be dumped whenever a jump instruction is executed. Tracing will begin with the first encounter of ALPHA and every fifth encounter until the 100th: BETA must be executed in order to increment this count. An octal dump with mnemonics and a console scoop will be given. JACK is written as identification on the standard output unit.

#### 4.3

### RECOVERY DUMP

On the RUN control statement (sec. 2.10) a recovery dump may be specified for abnormal termination of a program.

Recovery dumps consist of:

#### Console scoops

Program area print lines containing an absolute octal address, an octal address relative to the beginning of the subprogram, and the contents of four words in octal with mnemonics.

Common area print lines containing an absolute octal address, an octal address relative to the beginning of the common block and the contents of five words in octal.

When a new subprogram or common block is encountered, its name is printed and the relative address reset to zero.

Additional lines of identical words are not printed.

Information on recovery dump diagnostics is in Appendix C.

#### 4.4

### MEMORY MAP

If a memory map is to be suppressed, this must be indicated in the RUN control statement (sec. 2.10). When debugging aids are used, a map is always given. The locations are given as six octal digits, the leftmost designating the bank. The memory map lists the absolute location of the following items:

- subprograms
- program extension areas
- labeled common
- numbered common
- entry points

Information on memory map diagnostics is in Appendix C.

---

The loader performs the following functions:

- Loads and links subprograms
- Detects errors and provides diagnostics
- Patches subprograms and labeled common
- Assigns program extension areas
- Selects banks

A program may be divided into several subprograms, each separately compiled or assembled. The loader links these subprograms to each other and to library subroutines by associating entry points with external symbols.

The central control routine of SCOPE transfers control to the loader for loading drivers and programs to be executed. When loading is completed, control returns to the calling program. All errors detected are written out as diagnostics on the standard output unit (Appendix C). The loader provides for patching subprograms and labeled common and assigns a program extension area if necessary.

## 5.1 LOADER OPERATIONS

As each subprogram is loaded into storage, the names and locations of all entry points are entered into a symbol table. External symbols are also stored in the symbol table and linked with their corresponding entry points. When the loading process is complete, indicated either by two consecutive transfer cards or by a RUN statement, the loader searches the SCOPE library directories for subroutines corresponding to the names of all undefined external symbols. If any undefined symbol is not the name of a library subroutine, a loader diagnostic is written on the standard output unit and the job is terminated. If an undefined symbol is the name of an entry point to a library subroutine, the library subroutine is loaded into storage and the loader records the transfer address and returns control to the calling program. Unless the loader was called by the LOADER request, control returns to the next control statement on INP. Only 77775<sub>8</sub> locations are loaded into a bank.

All programs loaded for execution must have at least one transfer address, the entry point to which control will be transferred to begin execution. A transfer address is contained on a named TRA card.

The FORTRAN compiler compiles a single TRA card for each subprogram. A name is generated on the TRA card for each subprogram beginning with the PROGRAM statement. The COBOL compiler compiles a single TRA card for each paragraph. In COMPASS each END statement becomes a TRA card with the transfer address included if it was given in the END statement. Only one transfer card is needed for each subprogram. The ALDAP compiler compiles a single TRA card (without a transfer address) for each independently compiled procedure. A TRA card with a transfer address is always generated for each ALDAP compiled program. The name on the TRA card is the program name, if given, or a name generated by ALDAP.

If two transfer addresses are encountered in loading subprograms for execution, control transfers to the second address. The first address is placed in the A register, bits 41-24 (bits are numbered from right to left beginning with zero).

More than two transfer addresses will terminate a job, and a loader diagnostic will be written on the standard output.

The cards which the loader encounters while loading and processing subprograms and library subroutines are listed below. Details about the loader cards are in section 5.6.

IDC	Subprogram Identification Card
EPT	Entry Point Symbol Table
BCT	Block Common Table
RBD	Relocatable Binary Subprogram Deck
EXT	External Symbol Table
LAT	Linkage Address Table
BRT	Bank Relocation Table
OCC	Octal Correction Card
TRA	Transfer Card
LCC	Loader Control Card

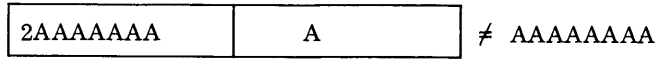
## LOADER NAMES

All loader names, including entry point names, external symbols, and subprogram names except common block names are 1-32 characters long. A name of 8 characters or less is contained in one word. If the name is greater than 8 characters but less than or equal to 31 characters, it is prefixed by 1-4, specifying the number of words comprising the name.

A name prefixed by a number is not identical to the same name not prefixed by a number.

**Example:**

1A ≠ A



first word

2nd word

1 word

**5.2  
LOADER CONTROL  
CARDS**

Loader control cards indicate subprograms to be overlaid, subprograms to be corrected, or the memory banks to which subprograms and common blocks are to be assigned. All loader control cards contain 11 (-), 0, 7, and 9 punches in card column 1. There are six types:

-  
0  
7  
9BANK,(b<sub>1</sub>), . . . ,name<sub>i</sub>, . . . (b<sub>2</sub>), . . . ,name<sub>k</sub>, . . .

or

-  
0  
7  
9BANK,(m),sym<sub>1</sub>,sym<sub>2</sub>, . . .

-  
0  
7  
9CORRECT,epname<sub>1</sub>,epname<sub>2</sub>, . . .

-  
0  
7  
9MAIN,u

-  
0  
7  
9OVERLAY,u,o

-  
0  
7  
9SEGMENT,u,n

} described in chapter 6.

**BANK  
STATEMENT**

There are two types of BANK statements.

The programmer may specify a particular bank for each subprogram and common block, or that particular subprograms and common blocks go into the same bank. This statement is placed before the subprograms to which it

pertains, if they are binary (object) subprograms; otherwise, it is placed immediately before the LOAD statement.

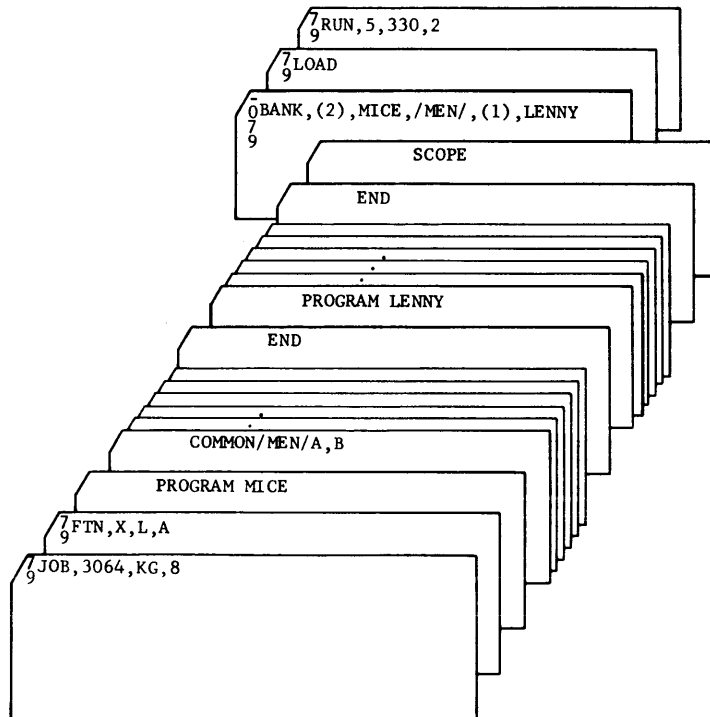
```
11
0
7BANK, (b1), . . . , namei, . . . , (b2), . . . , namek, . . .
9
```

b a bank number (0-7), an entry point, or a common block name.

name an entry point or common block name. A common block name is enclosed in slashes.

If b is an entry point or common block name, the names which follow it are allocated to the same bank as the entry point or common block name, and the loader places the subprograms in the bank having the largest amount of available storage, other than bank zero. If there are several entry points in a subprogram, only one of these need appear in the BANK statement.

Programs compiled or assembled by systems such as FORTRAN, COMPASS, ALGOL, must have provisions for bank relocation before they may appear in a BANK statement.



Two FORTRAN subprograms are to be compiled and written on the load-and-go unit. The BANK statement precedes the LOAD statement. Subprogram MICE and the common block MEN are to be placed in bank 2 and subprogram LENNY is to be placed in bank 1.

Various combinations of subprograms or common blocks may be forced into a particular bank.

```

11
0
7
9BANK, (m1),sym1,sym2, . . . ,(m2),sym3,sym4, . . .

```

$m_i$  is a bank number, 0-7

$sym_i$  may be the following designators:

SP. = subprograms	}	on binary input unit
NC. = numbered common		
LC. = labeled common		
LSP. = library subprograms	}	from library subroutines
LNC. = library numbered common		
LLC. = library labeled common		
APC. = SP. + NC. + LC.		
ALC. = LSP. + LNC. + LLC.		
ALL. = APC. + ALC.		

The designated subprograms and common blocks will be allocated to the specified bank. These declarations apply only to subprograms or common blocks for which no previous bank declaration defining a unique bank has been given.

**Examples:**

```

-
0
7
9BANK, (0), APC., LSP.

```

The succeeding subprograms, labeled, and numbered common, read from the binary input unit and library subprograms will be stored in bank 0. Numbered and labeled common blocks from the library are dynamically assigned by the loader.

```

-
0
7
9BANK, (A), B

```

-  
0  
7BANK,(0),A  
9

-  
0  
7BANK,(1),ALL.  
9

Subprograms containing entry points A and B will be forced into bank 0 by the first two bank statements. The remaining subprograms and common blocks will be loaded into bank 1.

## CORRECT STATEMENT

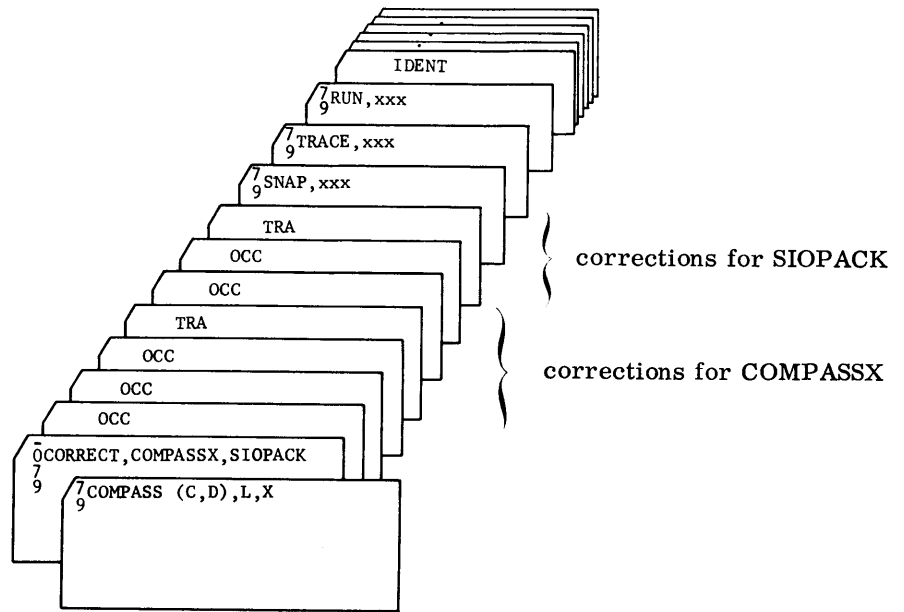
If library subprograms other than those specified on the entry point name statement are to be corrected, the CORRECT card is used.

11  
0  
9CORRECT,epname<sub>1</sub>,epname<sub>2</sub>, . . .

epname<sub>i</sub> is an entry point name within the library subprogram to be corrected; names must occur in the order in which they appear on the library tape.

Octal correction cards containing the corrections to the subprogram must follow the CORRECT card. The corrections for each subprogram are terminated by a single TRA card. The octal corrections must appear in the same order as the subroutines on the library tape to which they apply.

SNAP and TRACE cards may follow the last TRA card. The corrected deck and/or SNAP and TRACE cards must be followed by a RUN card. In this example COMPASSX must appear on LIB before SIOPACK.

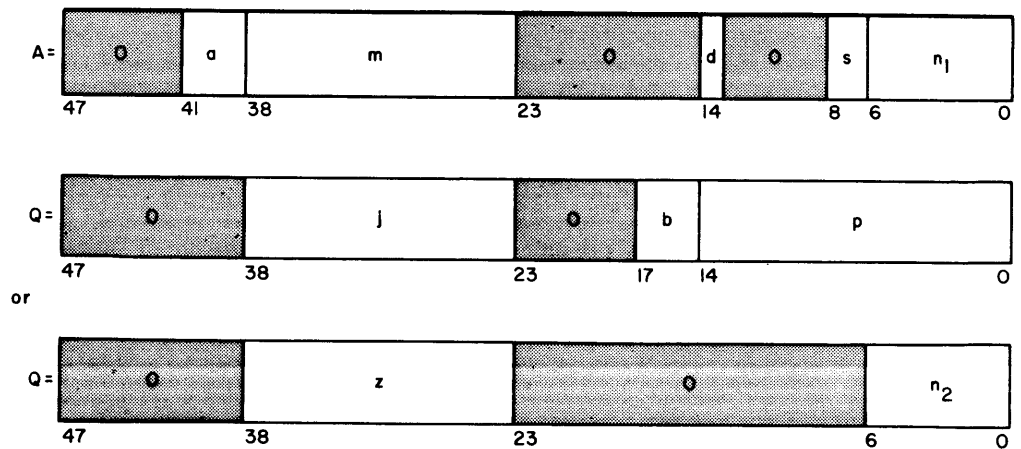


### 5.3 LOADER CALLS

When the LOADER request is made, the A and Q registers must contain certain parameters. The calling sequence is the following:

```
RTJ  LOADER
+    return
```

The parameters supplied in A and Q are:

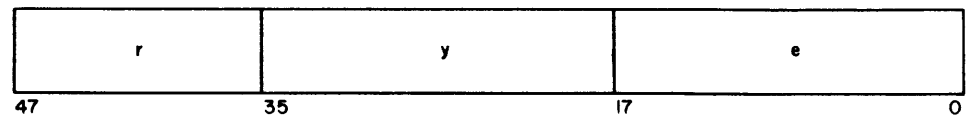




- am is an 18-bit address, or zero, specifying a location for preset entry points which have been defined by the calling program:
- d is a mapping parameter:
- 0 no map after loading
  - 1 map after loading
- s specifies the kind of loading operation:
- 00 load library programs from  $n_1$  (upper Q is used)
  - 01 load program from  $n_2$  (lower Q is used)
  - 10 load library program from  $n_1$ , and octal corrections to it from INP (upper Q is used) <sup>1</sup>
  - 11 complete loading operation after interruption - (ignore remainder of A and Q)
- $n_1$  the logical unit number of the library unit (usually 70).
- j the number of names in the list starting at bp.
- bp an 18-bit address specifying the beginning of a list of library subroutine entry point names to be loaded from the library tape ( $n_1$ ). A name may be in either form allowed for loader names (5.1).
- z the location, in bank zero, of the first binary card image of the program to be loaded. The rest of the cards are found on  $n_2$ . If  $z = 0$ , the first card is also found on  $n_2$ .
- $n_2$  designates the logical unit from which binary cards, or images of cards, one per record, are to be loaded.

The unused portions of A and Q must be zero.

The location defining the preset entry points contains:



- r a 12-bit value specifying the number of entry points which are preset into the entry point symbol table.
- y the first word address (18 bits) of the list of entry point names. The names must be in contiguous storage locations and may extend to 32 characters per name.
- e the first word address (18 bits) of the list of entry point addresses. This list must be contiguous words with one address occupying one word. Each address is contained in

the lower 18 bits of its word. The number of entry point addresses must equal the number of entry point names, each of which must equal  $r$ .

The four loading operations, keyed by the  $s$  parameter in the call to the loader, are as follows:

$s = 00$

Load from unit  $n_1$  the library subroutines called by the entry point names at  $bp$ . The entry point symbol table is preset if  $am$  is not zero.

SCOPE uses this call for COMPASS, COBOL, FORTRAN, ALGO, ALDAP and those programs defined on an entry point name statement, unless SCOPE parameters are used.

When the loader returns control to its calling program, the lower address of the A register specifies the first location in bank zero of the address list. This list gives the relocated addresses of the entry point names found at  $bp$  in the lower 18 bits of consecutive words. Bit 47 of the A register is zero. Bit 46 of the A register specifies whether numbered common in bank zero has (1), or has not (0) overlaid the loader. The Q register contains the number of errors encountered in loading, right justified. If a transfer address was encountered during loading, it is placed in the lower address of A, and the location of the address list is placed in the upper address of A.

Any time numbered common overlays the loader, the loader must be reloaded (Sec. 3.1 LOADER) before another loader operation can be executed.

$s = 01$

Load subprograms from unit  $n_2$ ; when two consecutive TRA cards are encountered, load externally-referenced library subroutines from  $n_1$ . If  $z$  is non-zero, it gives the location of the first card image in bank zero. An entry point symbol table is preset if  $am$  is non-zero.

SCOPE uses this call to load from INP or a load-and-go unit.

No more than two TRA cards may specify transfer addresses. When the loader returns to its calling program, the lower address of the A register gives the relocated address of the last transfer address encountered. If there were two transfer addresses, the first is given in bits 41-24 of the A register. Bit 47 of the A register is zero. Bit 46 of the A register specifies whether numbered common in bank zero has (1), or has not (0) overlaid the loader. The Q register contains the number of errors noted during loading, right justified.

If an end-of-file is encountered on  $n_2$ , control is returned with a 1 in bit 47 of the A register and bits 14-0, zero. If a binary card with  $w = 0$  is encountered, bit 47 is one, and bits 14-0 in A contain the location in bank zero of that card image. In both cases, library subroutines are not loaded before returning. Loading is completed by the  $s = 11$  call; loading is continued by another  $s = 01$  call.

s = 10

This is the same as the loading operation when s = 00, except that j must be 1 and OCC cards are accepted. After the named program from n<sub>1</sub> is loaded, OCC cards are loaded from INP until a TRA card is encountered. If a SCOPE control card is detected on INP, the loader returns with a 1 in bit 47 of the A register and bits 14-0 specify the location of the card image. Loading is completed by an s = 11 call.

SCOPE uses this call when loading programs defined by an entry point name statement with SCOPE parameters.

s = 11

Complete the loading operation after an end-of-file or SCOPE control card interruption. Only the s field in the A register is interpreted. If the previous operation was s = 01, library subroutines are loaded and return is made to the calling program.

If the previous operation was s = 10, processing of the octal corrections is completed, any remaining library subroutines are loaded, and return is made to the calling program.

#### 5.4 PROGRAM ASSIGNMENT

During loading, the program is assigned to various portions of storage. Programs which are too large for available storage may be divided into sections and executed sequentially under OVERLAY control. (Chapter 6).

#### BANK ASSIGNMENT

If the programmer specifies a particular bank, SCOPE loads the subprogram or common block into that bank. If the programmer specifies that particular subprograms and common blocks go into the same bank, as each is encountered it will be assigned to the bank with the most available storage, excluding bank zero. If there is no bank declaration, SCOPE selects the bank, other than zero, having the amount of available storage into which the subprogram or common block fits most tightly. Bank zero is assigned only when the other banks cannot provide space. Banks may be specified with the BANK statement (Sec. 5.2).

#### STORAGE ALLOCATION

After each load operation, SCOPE records the consecutive storage locations which have not been assigned to a subprogram, common block, or monitor routine. These constitute available storage and the limits may be obtained or changed by a programmer request during execution (MEMORY request, 3.6). Storage in a bank extends from the lowest location, 00001<sub>8</sub>, to the highest location, 77776<sub>8</sub>.

SCOPE resides in lower memory of bank 0.

Object subprograms, labeled common, program extension areas, programmer I/O drivers, and library subroutines are loaded into the highest available locations in the bank assigned.

Numbered common begins at the lower end of available storage in the assigned bank. In bank zero, numbered common overlays the loader. If numbered common is not in bank 0, the loader remains in storage. The last numbered common block in each bank may vary in size from one declaration to the next. Both numbered and labeled common blocks may be reduced in size from one declaration to the next.

The loader permits subprogram and labeled common modification by octal correction cards. Instructions which do not fit a corrected area in the program are loaded into a program extension area declared by the programmer and assigned by the loader. The program extension area is limited in size only by the amount of available storage in the bank into which the program and corrections are loaded.

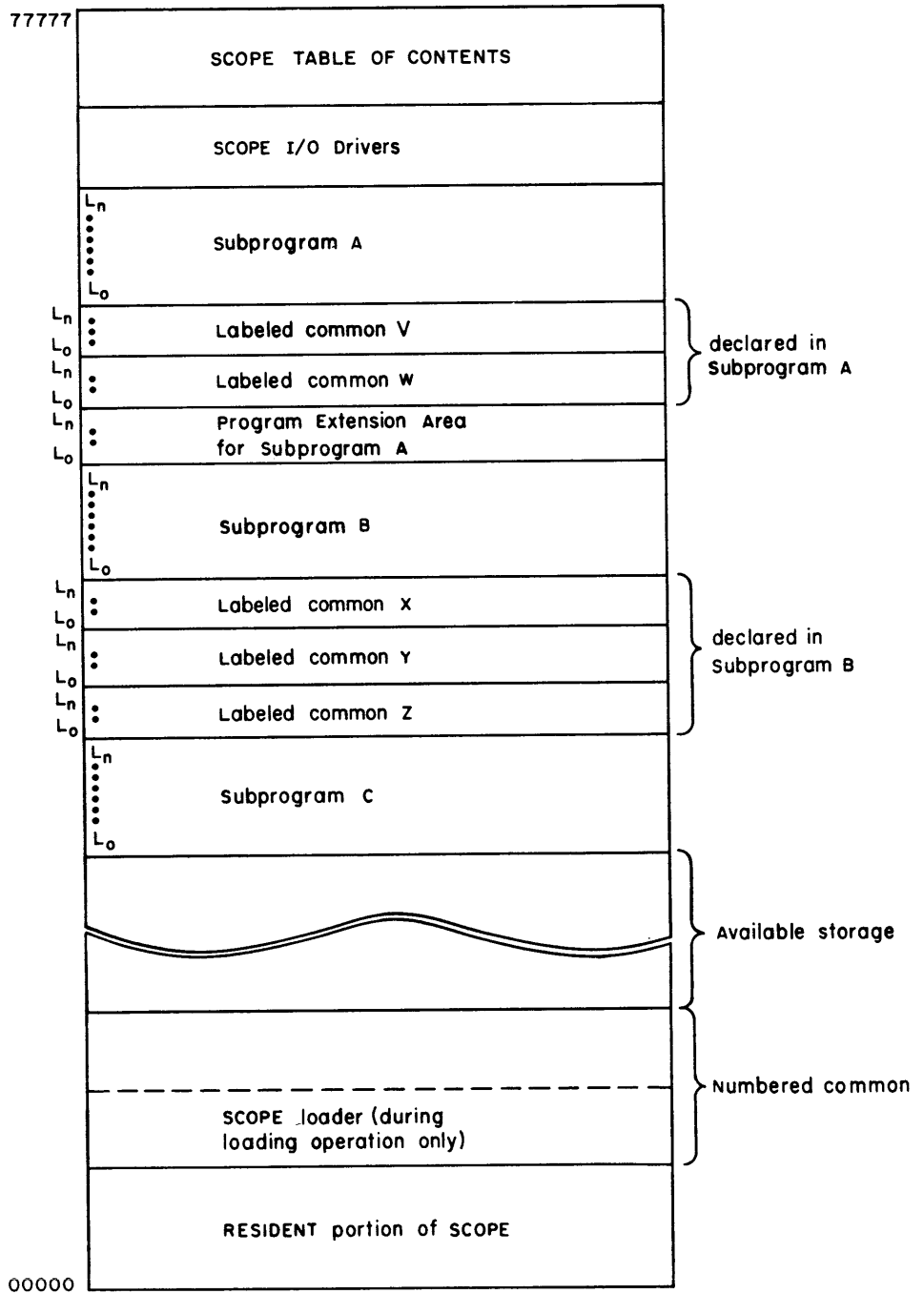
The locations occupied by a program after it is loaded can be obtained from the memory map, which includes a listing of the names and locations of all subprograms, labeled common, numbered common, program extension areas, and entry points in storage.

## STORAGE DIAGRAM

The following diagram illustrates how an object program might be stored.

Assuming that there is only one bank of storage, the SCOPE I/O drivers are loaded into the highest numbered area while SCOPE and the loader occupy the lowest portion of storage. During execution SCOPE and the I/O drivers are bounds protected.

The first subprogram, A, is loaded into highest available storage followed by labeled common blocks V and W. The loader is assigned the next available locations as a program extension area; the size is determined by information on the octal correction cards. Next, subprogram B is loaded; then labeled common X, Y, Z, and subprogram C. The area for numbered common overlays the loader; numbered common can not be preset. Subprogram C may or may not have contained octal corrections; however, if octal corrections were present, no program extension area was defined.



5.5  
**CORRECTING  
 SUBPROGRAMS**

Octal correction cards can be included with an object program at load-time. Existing instructions and labeled common data may be altered; additional instructions and data may be loaded into a program extension area. Any section of a program may be corrected. This area, determined and assigned by the loader, is limited in size only by the amount of available storage in the bank into which the program and corrections are loaded.

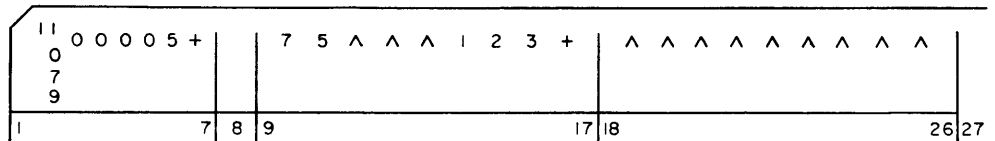
All octal correction cards contain a load address, the contents of one to four computer words, and relocation designators for the address portion of each instruction:

<u>Column</u>	<u>Contents</u>
1	punches in rows 11, 0, 7, and 9
2-6	relocatable load address, aaaaa, for the first correction field on the card
7	relocation factor for address aaaaa
8	blank
9-17	data field 1 - upper instruction or data word to be loaded at the address aaaaa
18-26	data field 2 - lower instruction or data word to be loaded at the address aaaaa
.	.
.	.
.	.
63-71	data field 7 - upper instruction or data word to be loaded at the address aaaaa + 3
72-80	data field 8 - lower instruction or data word to be loaded at the address aaaaa + 3

Octal correction cards are placed immediately before the TRA card of the binary subprogram to which they pertain.

**Example:**

To correct a single instruction in a subprogram:



In the fifth instruction of the subprogram, the upper instruction will be changed to an SLJ to relocatable 123 in the subprogram. The lower instruction will not be affected.

## RELOCATION FACTORS

The relocation factor, which follows the load address in card column 7, may be any one of the following:

<u>Factor</u>	<u>Relocation</u>
E	Relative to the first location of the program extension area
+	Relative to the first location of the subprogram
1	Relative to the first location of the first declared common block
2	Relative to the first location of the second declared common block
.	.
.	.
.	.
9	Relative to the first location of the ninth declared common block
0	Relative to the first location of the tenth declared common block

Only labeled common blocks may be corrected, data cannot be prestored in numbered common blocks. In selecting the correct factor for a common block, however, both numbered and labeled common blocks in the program must be counted in the order in which they are declared. Only the first ten blocks can be corrected.

### Examples:

A FORTRAN program contains the following statement:

```
COMMON /1/A/B3/G,H/B4/F/6/Z/COG/P
```

To alter data in block B3, the relocation factor 2 would be used. To alter data in block COG, factor 5 would be used. Data cannot be prestored in the numbered common blocks.

A COMPASS program contains the following statements:

```
21      BLOCK 10
        COMMON AFLAGS(5), BFLAGS(5)

H30     BLOCK 200
        COMMON A(10, 10), B(10, 10)

26      BLOCK 3
        COMMON R1, R2, R3

RTABLE  BLOCK 100
        COMMON R(10, 10)
```

To alter data in block H30, the relocation factor 2 would be used. To alter data in block RTABLE, the relocation factor 4 would be used. Data cannot be prestored in blocks 21 and 26 because they are numbered common.

## DATA FIELDS

The format of each data field, card columns 9-80, is:

nnnxxxxxi

Data fields 1-8 are loaded into sequential half-words in storage starting with address aaaaa (columns 2-6).

nnn the upper 9 binary digits of an instruction or data word.

xxxxx the address of an instruction, or lower 15 binary digits of a data word.

i the relocation factor for the address, xxxxx. Any of the factors in the relocation list may be used with two additions:

<u>Factor</u>	<u>Relocation</u>
blank	no relocation
-	Relative to the complement of the first address of the subprogram.

Since program instructions may refer to both numbered and labeled common, the value of the relocation designator, i, must be determined by counting each declared common block -- both labeled and numbered -- up to the one to which the address refers.

Blanks in the nnn and xxxxx fields are converted to zeros; if the entire field is blank, the related portion of storage is not altered.

## PROGRAM

**EXTENSION AREA** The size of the program extension area is defined by the largest reference to the area in the load address. Data field references are not taken into consideration when determining the size. If . . . . 2E were the largest load address, the program extension area would consist of three words (0, 1, and 2).

The programmer must provide a jump to the program extension area from the program to execute the octal corrections. A return jump from the program extension area to the program must also be included.





<u>Mnemonic</u>	<u>Name</u>	<u>Word Count</u> or <u>Card Type</u>
IDC	Subprogram Identification Card	31 <sub>8</sub>
EPT	Entry Point Symbol Table	32 <sub>8</sub>
BCT	Block Common Table	33 <sub>8</sub>
RBD	Relocatable Binary Subprogram Deck	1 ≤ W ≤ 21 <sub>8</sub>
EXT	External Symbol Table	34 <sub>8</sub>
LAT	Linkage Address Table	35 <sub>8</sub>
BRT	Bank Relocation Table	36 <sub>8</sub>
OCC	Octal Correction Cards	30 <sub>8</sub>
TRA	Transfer Card	37 <sub>8</sub>
LCC	Loader Control Cards	30 <sub>8</sub>

Card types 1 to 9 must occur in the order listed; only IDC and TRA are required in all subprograms. Type 10 occurs only between subprograms or ahead of the first subprogram in a series.

Unless identified as octal or coded decimal, information is assumed to be in binary on cards or in card format. In most cases more than one entry is contained on a card. With the exception of the octal correction card and the named TRA card, subprogram cards are punched in binary.

Each column on a card represents 12 bits of a 48-bit computer word. The correspondence between card positions and computer word bit positions for each group of four card columns is shown below.

<u>Corresponding Bit Position</u>				
<u>Row</u>	<u>Column 1</u>	<u>Column 2</u>	<u>Column 3</u>	<u>Column 4</u>
12	47	35	23	11
11	46	34	22	10
0	45	33	21	9
1	44	32	20	8
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
9	36	24	12	0

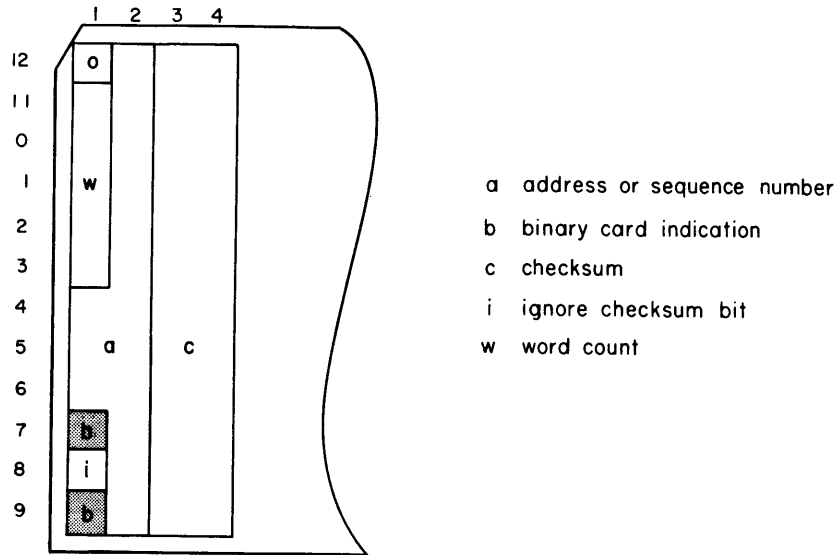
All SCOPE loader cards have a 7, 9 punch in column 1. On most loader cards the first four columns identify the type to provide a means of checking its contents.

<u>Mnemonic</u>	<u>Row</u>	<u>Column</u>	<u>Corresponding Word Bits</u>	<u>Purpose</u>
(none)	12	1	47	must = 0
w	11 3	1 1	46-42	card type or word count (RBD)
a	4 6	1 1	41-39	relocatable address (RBD)
	12 9	2 2	35-24	sequence number, or blank
b	7 9	1 1	38 and 36	indicates a binary card
i	8	1	37	indicates whether a checksum will be processed 1, checksum is ignored 0, checksum is compared
c	12 9	3 4	23-0	24-bit checksum

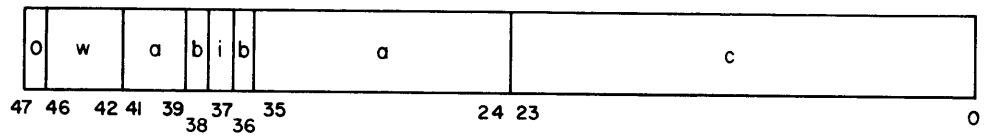
The remaining words (columns) depend upon the card type.

The checksum is formed by finding the 48-bit arithmetic sum of the 20 words on the card (columns 3 and 4 are set to 0 and left carries go into the low order bit). The high order 24 bits of the 48-bit sum are added to the low order 24 bits to form a 24-bit result. A left carry is generated if overflow into the 25th bit occurs. The 24-bit result is the checksum.

Card representation of first word:



Bit structure of first computer word:



## IDC CARD

The subprogram identification card names the subprogram which follows it and provides information about the subprogram to the loader.

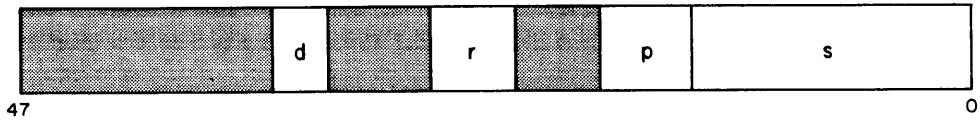
Card Content:

<u>Columns</u>	<u>Computer Word</u>	<u>Use</u>
1-4	1	Card type
5-8	2	Subprogram identification
9-24	3-6	Subprogram name in BCD
25-80	7-20	Zero

Word Content:

Word 1:  $w = 31_8$   
 a zero  
 c checksum

Word 2:



- d word number, 3-6, of the first word on the card which contains loadable data. Data begins in the (d+1)st word of all RBD cards.
- r length, 2 to 8, of the relocation byte on the RBD cards.
- p the number, 6 to 24, of relocation bytes per word on the RBD cards.
- s length of the subprogram in binary; 0 to  $77777_8$ .  
s is 0 if only common blocks follow.

The unused portions of word 2 are zero.

Words 3-6: name of subprogram in BCD, in format specified for loader cards (Section 5.1).

**EPT CARD**

The Entry Point Symbol Table defines the names and values for the entry points in the subprogram.

Card Content:

<u>Columns</u>	<u>Computer Word</u>	<u>Use</u>
1-4	1	Card type
5-80	2-20	These columns contain entry point names, in BCD, and related addresses, in binary.

Word Content:

Word 1:  $w = 32_8$   
 a sequence number in binary; all cards must be in sequence.  
 c checksum

Words 2-20:

Each entry point requires from 2 to 5 whole words. The name of the entry point occupies the first 1 to 4 words depending upon the length of the name. A name can be 1-32 characters, 6 bits per character. See section 5.1 for entry point name format. The address occupies positions 14-0 of the last of the 2 to 5 words. Each entry point definition immediately follows the preceding one; definitions continue from word 20 of one card to word 2 of the next card.

As many EPT cards as are necessary may be used to define the entry points of a subprogram; the cards must be in sequence.

## BCT CARD

The Block Common Table defines the name and length of each common storage area declared in the subprogram.

Card Content:

<u>Columns</u>	<u>Computer Word</u>	<u>Use</u>
1-4	1	Card type
5-8	2	Zero
9-80	3-20	Common declarations

Word Content:

Word 1:  $w = 33_8$   
a ascending sequence number in binary; cards must be in sequence  
c checksum.

Words 3, 5, 7, . . . 19: Names, in BCD code, of the blocks of common assigned. A name may not contain more than 8 characters; it may be numeric (for numbered common) or alphanumeric (for labeled common).

Words 4, 6, 8, . . . 20: Length of the common block named in the preceding word.

Common blocks are data storage areas which are shared by subprograms and library subroutines. Common may be labeled or numbered. For labeled common, the first character must be alphabetic. The 8-character name for a block of labeled common is assigned from high order to low order storage, following the subprogram in which it was first referenced. The 8-character name for a block of numbered common must be all numeric.

Numbered common is always assigned to locations in lower storage following SCOPE. Numbered common may not be preset; it may be assigned to the area used by the SCOPE loader and overlays the loader when it is referenced.

Subprograms sharing a block of common must each identify common by the same name and block length. There is one exception: the last numbered common block in a storage bank may vary in length between subprograms.

Up to 126 blocks may be defined for one subprogram.

## RBD CARD

The Relocatable Binary Subprogram Deck contains the instructions, constants, and data to be positioned. A relocation increment or decrement may be applied to the address portions of each word.

### Card Content:

<u>Columns</u>	<u>Computer Word</u>	<u>Use</u>
1-4	1	Number of instruction words on the card, the first relocatable address and a checksum.
5-24	2-6	Relocation bits, in constant length bytes, which determine the kind of relocation and the address portions of each word. Up to 6 computer words are filled.
13-80	4-20	Instruction words to be loaded. As many computer words are used as are available.

### Word Content:

Word 1:     w = word count, 1 to  $21_8$   
               a   initial load address for the words on the card  
               c   checksum  
               b, i are as defined for all standard card types

The number of words reserved for the relocation bytes depends on the length of each byte (2-8 bits) which in turn depends on the number of common blocks to be used. The number of words may be determined from the following table:

No. of Common Blocks	Relocation Byte Length	Max. No. of Bytes/Word	Total Possible No. of Bytes	No. of Words for Relocation Bytes
0	2	24	35	2
1-2	3	16	33	3
3-6	4	12	33	3
7-14	5	9	31	4
15-30	6	8	31	4
31-62	7	6	29	5
63-126	8	6	29	5

An integral number of bytes is arranged in a word, left justified. The first byte in word 2 designates relocation for the load address. Subsequent bytes correspond to left, then right, address portions of the first instruction word, then the second, and so on. The value of the byte, excluding the leftmost bit, is used to locate an entry in a table of relocation factors during loading (RFTABLE). The leftmost bit of the byte specifies incrementing (0) or decrementing (1) the contents of the related portion of the instruction or the load address. The remaining bits give the ordinal of the proper word in the table. The first word of this table contains the relocation factor for fixed addresses and, therefore, contains zero. The second word contains the factor for addresses to be relocated relative to the subprogram; the third word for addresses to be relocated relative to the first declared common block on the BCD card; the fourth word for the second declared common block, and so on. If a byte points to an undeclared common block, an error diagnostic is printed.

If an address is to be decremented, the complement of the indicated relocation factor is used when relocating the address. The words not used by the relocation bytes are used for the relocatable instruction, constants, data, and so forth. These words always begin in the same column, for each subprogram, as though the maximum number of relocation bytes were needed.

A byte length, the word number of the last word containing relocation bytes, d, and the total number of bytes per word, p, are contained on the IDC card.

A byte value of 10 . . . 0 is illegal.

A byte value of 00 . . . 0 implies that the address is not to be modified.

A byte value of 0 . . . 01 or 10 . . . 01 implies a subprogram instruction.

A byte value of 0 . . . 010 or 10 . . . 010, refers to the first common block; 0 . . . 011 or 10 . . . 011, the second; 0 . . . 100 or 10 . . . 100, the third, and so on. The reference to the 126th common block would be 01111111 or 11111111.

The load address byte may not be 0 . . . 0, or 1x . . . x.



## EXT CARD

The External Symbol Table contains names of the symbols external to the subprogram. The names are entry points to other subprograms and library subroutines.

### Card Content:

<u>Columns</u>	<u>Computer Word</u>	<u>Use</u>
1-4	1	Card type.
5-80	2-20	External symbol names in BCD.

### Word Content:

Word 1:  $w = 34_8$   
a ascending sequence number in binary  
c checksum

Words 2-3: External symbol names in BCD are arranged contiguously on a card and may be split between successive cards; a name may continue from column 80 of one card to column 5 of the next card. Loader name format is given in Section 5.5. As many EXT cards as are necessary may be used to define external symbols; the cards must be in sequence.

## LAT CARD

The Linkage Address Table provides linkage for all references to external symbols. A minimum of one LAT word exists for every external symbol.

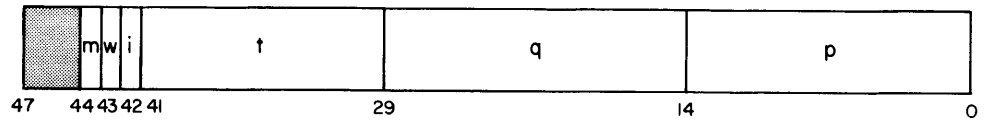
### Card Content:

<u>Columns</u>	<u>Computer Word</u>	<u>Use</u>
1-4	1	Card type.
5-80	2-20	Relocatable addresses and related information for the external symbols referenced in the subprogram.

### Word Content:

Word 1:  $w = 35_8$   
a ascending sequence number in binary  
c checksum

Word 2-20: Each LAT entry, one word in length, points to a single string of references to one EXT symbol. The LAT entries for each EXT symbol are threaded through the table.



- m relocation mode indicator for the external symbol.
  - 1 normal (incremented) references
  - 0 complementary (decremented) references
- w upper/lower string indicator.
  - 0 string starting at p refers to an external symbol in the upper portion of the word in the subprogram.
  - 1 string starting at p refers to an external symbol in the lower portion of the word in the subprogram.
- i 1
- t ordinal of the next LAT entry related to the same external symbol or zero if no more LAT entries are needed.
- q is the quantity added to the relocated address of the EXT symbol (q may be zero).
- p relocatable address of the word in which the external symbol is used. It is the initial location in the subprogram of the string of references to an EXT + Q. This address gives the location of the next reference, and so on. The last address in the string contains  $77777_8$ .

For each EXT symbol there will be at least one LAT entry. If the identical symbol appears in both the upper and lower positions of a word, there will be two LAT entries. There will be other LAT entries for each modification value, q, of the symbol. Finally, there will be separate entries for each q-valued portion of the symbol for either value of the relocation mode indicator, m. There may be many LAT entries for a single external symbol. For example, each of the following references will require two LAT entries if each is referenced from an upper and lower portion of a word. Eight entries would result from:

- EXT + Q<sub>1</sub>
- EXT + Q<sub>2</sub>
- EXT + Q<sub>1</sub>
- EXT + Q<sub>2</sub>

The EXT and LAT tables are arranged in parallel so that the *ith* LAT begins the series of references to the *ith* symbol. When more than one LAT entry exists for an EXT symbol, successive entries continue after all initial LAT entries. The related LAT entries are threaded together by the *t* designation in the entry.

If a symbol with unique values for *m*, *w*, and *q* appears only once in a subprogram, the address of the word using it is recorded in the *p* portion of the entry. If the same symbol with identical values of *m*, *w*, and *q* appears more than once in a subprogram, SCOPE assumes that all these references are strung together. The string is constructed so that the address portion of the word in which the particular version of the symbol is used is replaced by the location of the word in which the next reference occurs. The location of the following reference is placed in the preceding address portion, and so on. The last address portion is assumed to contain 77777<sub>8</sub>. The location of the first word of the string is recorded in the *p* portion of the LAT entry. This method of stringing identical versions of the symbol allows SCOPE to replace the relocatable address by a particular relocated address for the symbol when it can be determined (defined as an entry point).

The designator, *t*, is used to locate the next LAT entry for a specific symbol regardless of the values of *m*, *w*, and *q*. The thread of LAT entries is terminated by *t* setting to zero. *t* is also zero if the symbol occurs only once.

If an extraneous EXT entry exists, (an EXT is listed but not referenced) the LAT entry will have the value: *m* = *w* = *t* = *q* = 0 and *p* = 77777<sub>8</sub>.

If no LAT entry exists for an EXT entry, it is an error.

As many LAT cards as are necessary may be used, they must be in sequence and all but the last must contain 19 entries.

SOURCE SUBPROGRAM			OBJECT SUBPROGRAM		
LOC	UPPER ADDRESS	LOWER ADDRESS			
a a a a a		SAM	a a a a a		7 7 7 7 7
b b b b b	SAM	-SAM-2	b b b b b	7 7 7 7 7	7 7 7 7 7
c c c c c		SAM+2	c c c c c		7 7 7 7 7
d d d d d		SAM+2	d d d d d		c c c c c
e e e e e		ROGER+5	e e e e e		7 7 7 7 7
f f f f f		ROGER+5	f f f f f		e e e e e
g g g g g		ROGER+6	g g g g g		7 7 7 7 7

EXT ENTRIES

SAM
ROGER
ANDY

LAT ENTRIES

ORDINAL	m	w	t	q	p
0	1	1	3	0	aaaaa
1	1	1	5	5	fffff
2	0	0	0	0	77777
3	1	0	4	0	bbbbbb
4	1	1	6	2	dddddd
5	1	1	0	6	gggggg
6	0	1	0	-2	bbbbbb

**BRT CARD**

The Bank Relocation Table indicates locations within the subprogram where bank designators depend on the banks to which this or other subprograms or common blocks are assigned. As many BRT cards as necessary may be used; they must be in sequence.

Card Content:

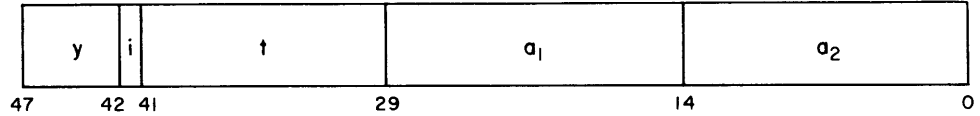
<u>Columns</u>	<u>Computer Word</u>	<u>Use</u>
1-4	1	Card type.
5-80	2-20	Threaded list of BRT entries.

Word Content:

Word 1:     w = 36<sub>8</sub>  
               a   sequence number in binary  
               c   checksum

Words 2 - 20: The BRT table has three sections. The first section parallels the EXT table and each word contains the first two entries for that EXT symbol. If no bank designators depend on that symbol, there are no entries. The second section consists of a pointer to the beginning of the thread of entries for this subprogram and each common block it defines. Section three holds the rest of the BRT entries, two per word, pointed to from sections one and two.

Format of the words in sections one and three:



$y$  specifies one of the five bank designator positions to be relocated within the word at  $a_1$  and  $a_2$ .

The value of  $y$  may be one of the 30 possibilities,  $1-36_8$ :

$$y = (\text{code for } a_1 \text{ designator}) + 5 \text{ times } (\text{code for } a_2 \text{ designator}).$$

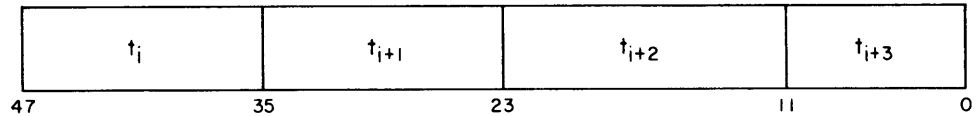
If  $a_2$  is not present,  $y$  is 1 to 5.

$i$  0

$t$  designates the next BRT word (two entries) in this thread.

$a_1$  and  $a_2$  are addresses in this subprogram in which a relocatable bank designator appears. If more than one designator in a word is relocatable, each has a BRT entry.

Section two parallels the RFTABLE with four entries per word:



$t_i$  is the ordinal in section three, relative to the beginning of BRT, at which the first two BRT entries corresponding to the  $i$ th RFTABLE entry occur.  $t_1$  points to the entries depending on the bank into which the subprogram is loaded.  $t_2$  through  $t_{127}$  (or the last  $t$  field) point to the entries depending on the banks to which the corresponding common blocks are assigned.  $t_0$ , and any fields not pointing to a section three entry are zero. As many  $t$  fields are required as the number of declared common blocks.

### TRA CARD

The Transfer Card signals the end of the subprogram deck or the entire program deck, and in some cases, the starting location of the program.

Card Content:

Columns	Computer Words	Use
1-4	1	Card type.
9-80	not applicable	Entry point name of the starting address of the program.

Word Content:

Word 1:      w = 37<sub>8</sub>  
                 a    0  
                 c    checksum

Beginning in column 9, an entry point transfer name is punched in Hollerith. Up to 31 Hollerith characters may be used. When there is no transfer name, columns 9 through 80 must be blank.

Usually, only one subprogram or library subroutine encountered in a loading operation specifies a transfer name. Control is given to the address specified by that name when the program is run. However, a second transfer name may be specified in a later subprogram or library subroutine. When the program is run, control is given to the second address, the first address is placed in the A register (bits 41-24). Except for overlays, it is an error to declare more than two entry point addresses.

## LCC CARD

The Loader Control Cards indicate overlay subprograms and the banks to which subprograms and common blocks are to be assigned.

Card Content:

<u>Columns</u>	<u>Computer Word</u>	<u>Use</u>
1      w = 30 <sub>8</sub>	not applicable	Card type 11, 0, 7, 9.
2-80	not applicable	Hollerith characters representing loader control statements and parameters.

In the six types of LCC cards, columns 2 through 80 are free field; parameters are separated by commas, and blanks are ignored. All information must be contained on one card.

MAIN,u  
OVERLAY,u,o  
SEGMENT,u,n }      see chapter 6

- u    decimal number, 0 to 50 or 69, specifying the unit on which the main program, overlay or segment is to be written.
- o    decimal number specifying the overlay number.
- n    decimal number specifying the segment number.

BANK, (b <sub>1</sub> ), . . . , name <sub>i</sub> , . . . , (b <sub>2</sub> ), . . . , name <sub>k</sub> , . . . )	}	see section 5.2.
BANK, (m), sym <sub>1</sub> , sym <sub>2</sub> , . . .		
CORRECT, epname <sub>1</sub> , epname <sub>2</sub> , . . .		

- b bank designator, 0-7; the name of an entry point, enclosed in parentheses (entry point); or the name of a common block, enclosed in slashes and parentheses (/block/).
- m bank designator, 0-7.
- name entry point or common block name; a block name must be enclosed in slashes.
- sym<sub>i</sub> symbol to specify subprogram, numbered or labeled common, or library subroutines.
- epname<sub>i</sub> entry point name in a library subroutine.

## OCC CARD

The Octal Correction Cards provide facility to correct existing instructions in the subprogram and to add new instructions in a program extension area preceding the subprogram in storage. These cards are not automatically produced by assembling or compiling a source deck; they must be provided by the user.

### Card Content:

<u>Columns</u>	<u>Use</u>
1     w = 30 <sub>8</sub>	Card type.
2-7	Relocatable load address for first correction field on card and relocation increment.
9-80	Corrections or additions.

Information on this card is punched in Hollerith characters to be loaded into sequential half words in storage. The card is divided into 9 fixed fields; the first consists of 6 characters; the remainder, 9 characters each. This card is described in Section 5.5.

# PREPARATION OF OVERLAY TAPES 6

---

Overlay processing allows programs that exceed available storage to be divided into independent parts which may be called and executed as needed. A program may be divided into a main section and any number of overlays, each of which may contain any number of segments. Main, overlay, and segment may each contain subprograms. Only main, one overlay, and one segment may occupy storage at a given time.

In a program containing overlays, the loader control statements, MAIN, OVERLAY, and SEGMENT, precede the relocatable binary subprograms which comprise the respective sections. After a source program is assembled or compiled, overlay processing loads the relocatable binary subprograms into storage and writes each overlay or segment as a separate record in absolute binary on an overlay tape. This overlay tape is then called in sections for execution. The absolute records do not require the relocatable binary loader to perform the usual relocating and linking functions. Debugging aids cannot be used.

Initially, control is transferred to main which resides in storage continuously; it in turn calls the overlays when they are needed during program execution. Segments may be called either by main or an overlay. FORTRAN and COMPASS subroutines are available to call the overlays and segments during execution; these must be included in the source subprograms. Once an overlay tape is created, it may be executed subsequently with the SCOPE control statement, LOADMAIN.

An overlay tape is composed of absolute binary records, followed by two end of files. Each record, constituting a main subprogram, overlay, or segment, may be composed of many subprograms. A particular main, overlay, or segment may occupy any number of banks when in absolute form.

## 6.1 LOADER CONTROL STATEMENTS

The overlay tape is prepared by preceding each section of the program with a loader control statement specifying whether it is a main section, overlay or segment. A loader control statement (as opposed to a SCOPE control statement) is interpreted by the loader and contains an 11, 0, 7, 9 punch in card column 1. Each statement specifies the logical unit on which the resultant absolute binary program will be stored. The cards following a loader control statement may consist of binary subprograms, compiler language or assembly language subprograms; SCOPE control statements may be included if compilation and assembly is to be performed prior to creation



of the overlay tape. Each main, overlay or segment must contain one transfer address. Main may contain two transfer addresses.

## MAIN

$\bar{0}$   
7  
9 MAIN,u

- u is the logical unit number of the overlay tape, 1-49, on which the main section of the program is to be stored in absolute binary. u may not be omitted.

The MAIN control statement may precede the object subprograms which comprise the main section; it defines the main part and the logical unit number of the overlay tape on which it is to be stored.

When overlay tapes are being prepared, the main section remains in storage for immediate execution. After all sections of the program have been stored on tape, the main section may be executed.

The main subprograms must precede the first overlay; if the MAIN statement is omitted, an overlay tape is written containing just the overlays and segments. The overlay tape may be used for immediate execution whether it contains main or not; however, if the overlay tape does not contain main, it cannot be used in subsequent executions.

Portions of the main section may be assigned to several banks with the BANK statement.

## OVERLAY

$\bar{0}$   
7  
9 OVERLAY,u,o

- u is a logical unit number, 1-49, of the overlay tape on which the overlay section is to be written in absolute binary. u may not be omitted.
- o is the decimal number identifying the overlay.

The OVERLAY control statement precedes the object subprograms which comprise the overlay; when it is encountered SCOPE creates an overlay section and writes this section in absolute binary on the overlay tape specified by the logical unit number.

The main section must include calling sequences to call each overlay into storage. The overlay may be assigned to several banks with the BANK statement.

## SEGMENT

<sup>0</sup>  
7  
9 SEGMENT,u,n

- u is the logical unit number, 1-49, of the overlay tape on which the segment is to be written in absolute binary. u may not be omitted.
- n is the decimal number identifying the segment.

The SEGMENT control statement precedes the object subprograms which comprise the segment; when it is encountered SCOPE creates a section and writes it in absolute binary on the overlay tape specified by the logical unit number.

The main or overlay section must include calling sequences for each segment. The segment may be assigned to several banks with the BANK statement.

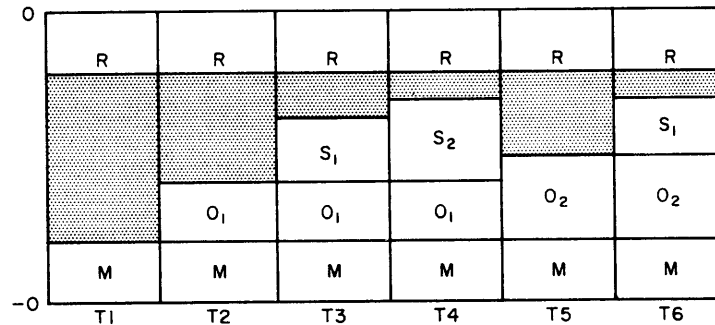
## RULES

Rules for partitioning a program into overlays and segments:

1. Numbered and labeled common and all entry points declared in the main subprograms may be referenced by any overlay and any segment.
2. Numbered and labeled common and all entry points declared in an overlay may be referenced by that overlay and its associated segments, but not by the main subprograms, another overlay, or segments contained in another overlay.
3. Numbered and labeled common and all entry points declared in a segment may be referenced by that segment only.
4. The first overlay card must be preceded by a main program. If the main program is not declared with a MAIN card, it is not written on the overlay tape.
5. The overlay numbers must start with one and be consecutive for all overlays written.
6. The segment numbers, within an overlay, must be consecutive starting with one.
7. Only four overlay tapes may be written. No overlay tape may occupy more than one reel.
8. Each overlay and segment must have a single named transfer point.
9. All segments for a particular overlay must immediately follow that overlay on the tape.

## STORAGE DIAGRAM

A diagram of storage during overlay processing and execution:



R = Resident  
M = Main  
O<sub>i</sub> = Overlay  
S<sub>i</sub> = Segment  
T<sub>n</sub> = Time n

## DECK STRUCTURE

Deck structure in the preparation of an overlay tape differs according to input. Relocatable binary subprograms, preceded on INP by the loader control statements, MAIN, OVERLAY, and SEGMENT, are loaded and then written on the overlay tape in absolute binary. Source language subprograms must be compiled/assembled onto a load-and-go tape before they can undergo overlay processing; the loader control statements MAIN, OVERLAY, SEGMENT, preceding each portion coded in source language, must be transferred to the load-and-go tape. The load-and-go tape then becomes input for the overlay tape. Relocatable binary subprograms and source language subprograms may be combined as input.

As the overlay tape is prepared, each section is assigned absolute locations in storage. As each loader control statement is read, it is written on OUT. The section following the loader control statement is loaded into storage and a MAP is written on OUT showing the absolute locations assigned to each main, overlay, and segment.

## BANK ASSIGNMENT

Any subprogram within a main section, overlay, or segment may be assigned to a specific bank. In source language subprograms, a bank may be selected by the appropriate source language pseudo instruction.

If relocatable binary subprograms are the input for an overlay tape, they may be assigned to specific banks by the loader control statement:

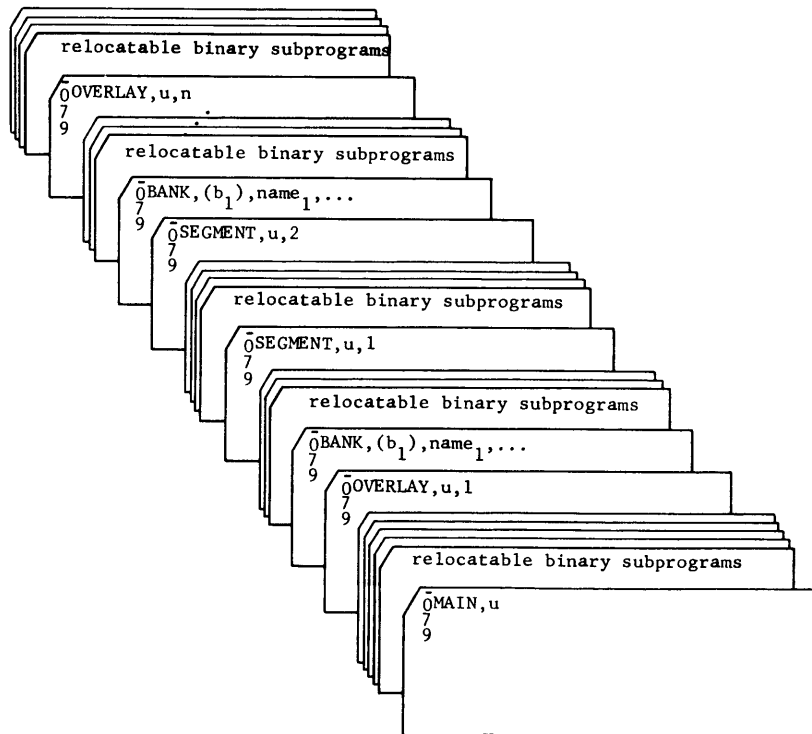
$$\begin{array}{l} \bar{0} \\ 7 \\ 9 \end{array} \text{BANK}, (b_1), \text{name}_1, \dots, (b_k), \text{name}_1, \dots, \text{name}_n$$

The BANK statements must precede the binary subprograms to which they pertain; however, BANK statements must follow the OVERLAY or SEGMENT statement to which they pertain. If the relocatable binary subprograms are contained on a load-and-go tape as the result of a compilation/assembly, the BANK statements may not precede the LOAD statement which loads the load-and-go tape for overlay processing.

If there are no BANK statements, normal bank assignment occurs.

**BINARY  
SUBPROGRAMS  
ONLY**

To create an overlay tape when the deck consists only of relocatable binary subprograms, each subprogram must be preceded by a control statement, MAIN, OVERLAY, or SEGMENT. If a subprogram is to be assigned to a specific bank, a BANK statement must be included before the binary subprogram and after the loader control card. The order in this example (MAIN, subprograms, OVERLAY, subprograms, . . . etcetera) must be followed in all overlay programs.



The overlay tape logical unit numbers may be the same or they may differ; however, segments must follow the overlay on the same overlay tape. If the information in the preceding example were on a tape unit other than INP, it could be loaded by a LOAD control statement and processed as though it were on INP. With the deck in the preceding example on INP, processing proceeds as follows:

1. Relocatable binary subprograms following the MAIN statement and up to the first OVERLAY statement are loaded into storage and linked by the loader. They are then written in absolute binary in a single record on the overlay tape, logical unit u. The first two words which identify it as the main record contain the absolute transfer address. The control statement, MAIN, is written on OUT followed by a MAP of the main section showing the assigned absolute locations. If the main section is not preceded by a MAIN statement, it is not written on the overlay tape, although it is retained in storage during overlay processing.
2. The binary subprograms following the first OVERLAY statement are loaded into an area beginning with the first location after the main area. They are written in absolute binary as a single record on the overlay tape, logical unit u. The first word of the record identifies it as an overlay with its overlay number (1 in the first case). The control statement, OVERLAY, is written on OUT followed by a MAP of the overlay section showing the assigned absolute locations. The BANK statement following the OVERLAY statement controls the assignment for subprograms contained in the first overlay.
3. The BANK statement following the second SEGMENT statement controls bank assignment for subprograms contained in segment 2 of overlay 1.

Step two is repeated for each overlay and segment. In storage, the segment begins with the first location after its related overlay. During this process the main section remains in storage.

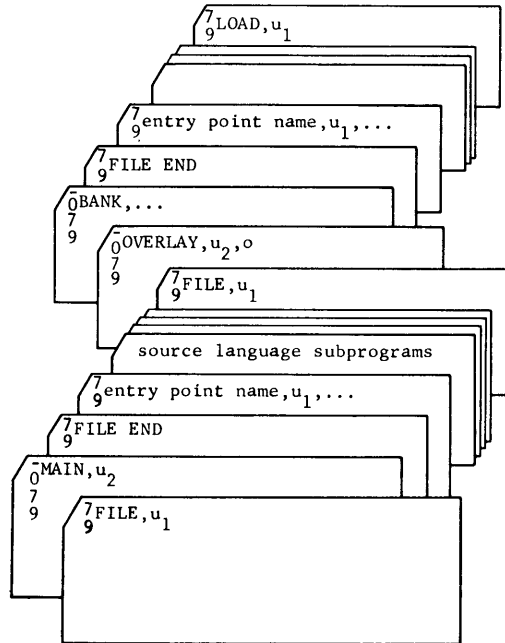
All the overlays are loaded into the same area following main before they are written on the overlay tape. All segments are loaded into storage beginning at the same first location following the associated overlay before they are written on tape.

4. The end of overlay processing is signaled by a SCOPE control statement such as  $\overset{7}{9}$ RUN, or two consecutive transfer cards.

#### **SOURCE LANGUAGE SUBPROGRAMS**

If source subprograms comprise the overlay deck, they must be compiled/assembled onto a load-and-go tape. Loading of the load-and-go tape for overlay processing is initiated by a LOAD statement. The loader control statements, MAIN, OVERLAY, and SEGMENT, must precede the subprograms on INP and

the load-and-go unit as usual; they are transferred to the load-and-go tape by FILE, FILE END control sequences. Bank assignment may be included in the source language subprograms; BANK statements may be transferred to the load-and-go tape by FILE, FILE END control sequences.  $\overline{7}_9$ COMPASS and  $\overline{7}_9$ FTN are entry point name statements; they must immediately precede the subprograms to be assembled or compiled. Entry point name statements must follow each FILE, FILE END sequence to return control to the compiler/assembler.



With this deck on INP, processing proceeds as follows:

1. The MAIN statement is transferred to the load-and-go unit,  $u_1$ .
2. The subprograms following the first FILE END statement will be processed by the named library program. The entry point name statement should specify that the binary object subprograms be stored on the load-and-go unit,  $u_1$ .
3. Steps 1 and 2 are repeated until the LOAD statement is encountered. At this time the load-and-go unit,  $u_1$ , contains the same deck structure as INP would have contained (Section 6.1) were only binary subprograms included in the deck, as follows:

$\overline{0}$   
 $\overline{7}_9$ MAIN,  $u_2$

relocatable binary subprograms

$\bar{0}$   
 $\begin{matrix} 7 \\ 9 \end{matrix}$ OVERLAY, $u_2, o$

$\bar{0}$   
 $\begin{matrix} 7 \\ 9 \end{matrix}$ BANK, . . .

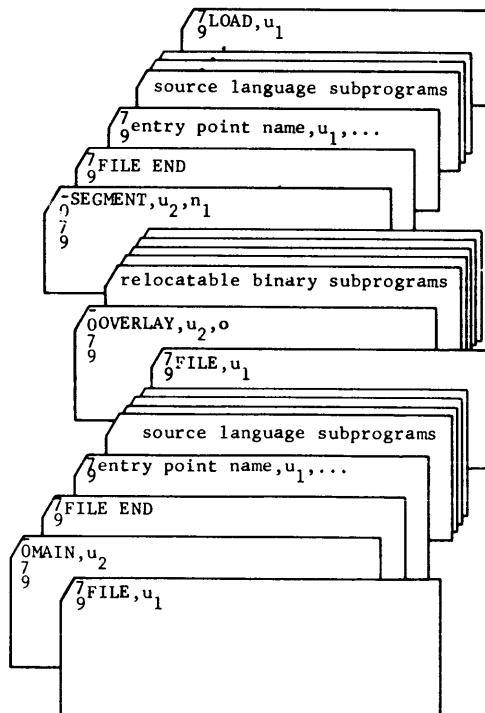
relocatable binary subprograms

.

- 4. When the LOAD statement is encountered, the load-and-go unit,  $u_1$ , is loaded, and processing is identical to the overlay processing for binary subprograms only. The overlay tape is on logical unit,  $u_2$ .

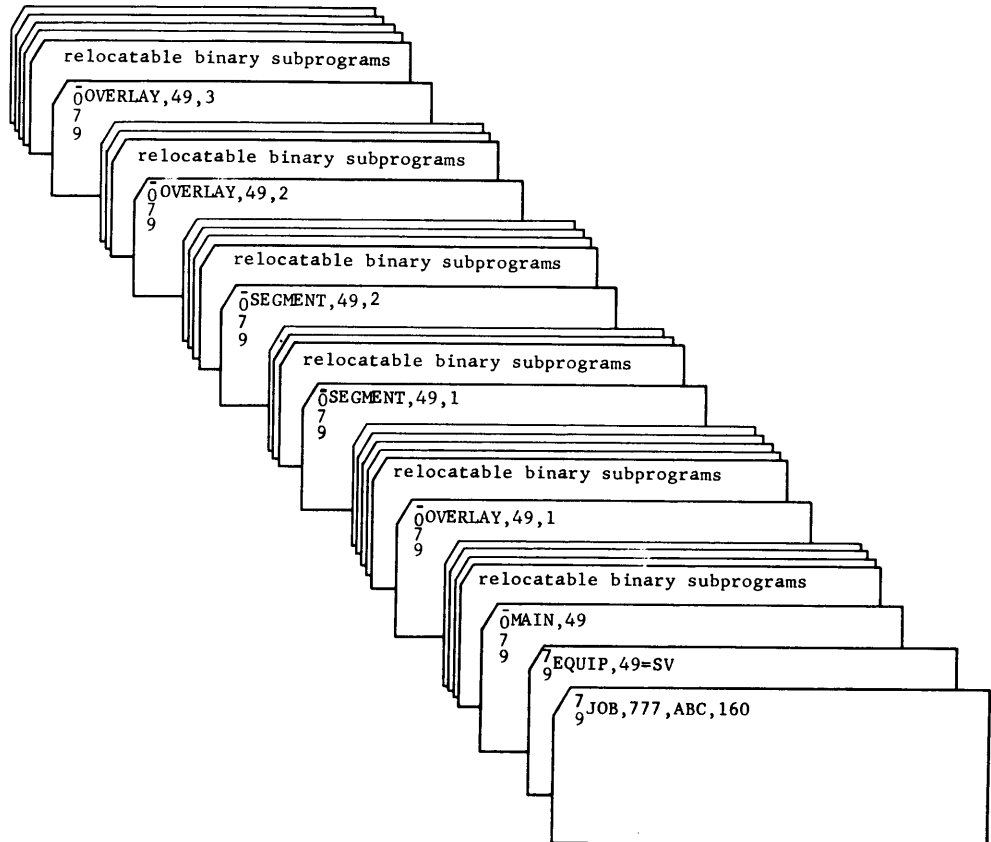
### BINARY AND SOURCE SUBPROGRAMS MIXED

If relocatable binary subprograms are mixed with source language subprograms in an overlay deck, the binary subprograms must be transferred to the load-and-go unit by FILE, FILE END sequences. The order on the load-and-go unit must be the same as that on INP in the example (Section 6.1). Processing is similar to that for source language subprograms.



## 6.2 EXECUTING OVERLAY PROGRAMS

When an overlay tape is prepared, it may be saved with an EQUIP control statement for subsequent executions. If the overlay tape is to be prepared for subsequent executions only, no RUN control statement is required.



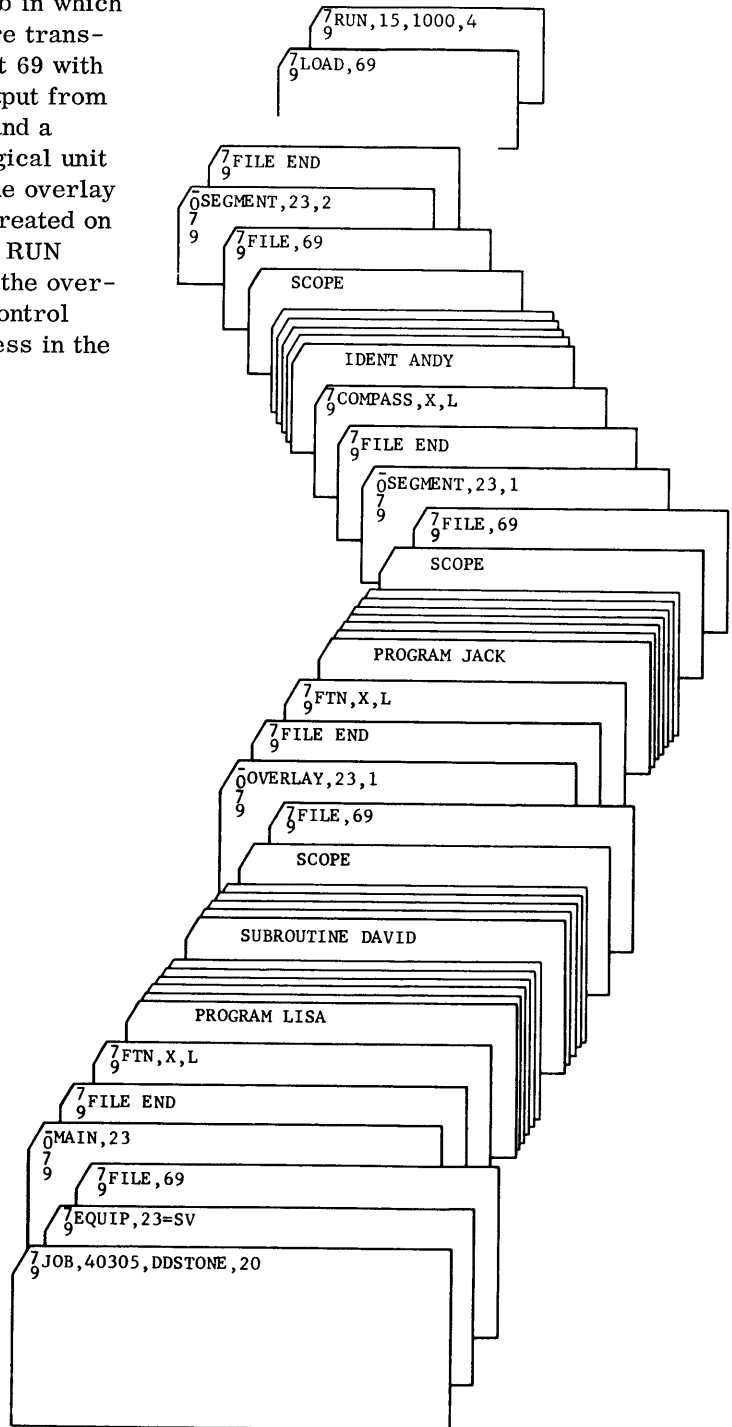
## IMMEDIATE EXECUTION

When the overlay tape is prepared, the main section is retained in storage. To execute immediately, a RUN control statement (LOADMAIN is not used) must be placed at the end of the overlay deck or the LOAD statement on INP. When the RUN statement is encountered, the overlay tape is rewound and control goes to the transfer address in the main section. The main section then calls the overlays and segments from the overlay tape.





- 2) This deck illustrates a job in which the overlay statements are transferred to load-and-go unit 69 with the relocatable binary output from FORTRAN compilations and a COMPASS assembly. Logical unit 69 is loaded to prepare the overlay tape; an overlay tape is created on logical unit 23. When the RUN statement is encountered the overlay tape is rewound and control goes to the transfer address in the main section.



**PREPARED  
OVERLAY TAPE**

Once an overlay tape has been prepared, the overlay program in absolute binary can be loaded and executed with the SCOPE control statement, LOADMAIN.

<sup>7</sup><sub>9</sub>LOADMAIN, u, t, p, r

t time limit in minutes (optional)

If t is blank or larger than the remaining job time limit, the job limit will be used.

p print limit (optional)

r request for octal dump or console scoop on abnormal termination

non-blank, octal dump of all core

blank, console scoop

LOADMAIN loads the main section from the overlay tape on logical unit u and transfers control to the transfer address in the main section. During execution the main section calls the overlays and segments from the overlay tape. LOADMAIN cannot be used unless the main section is included on the overlay tape. A RUN control statement is not required to execute overlays which are loaded by LOADMAIN.

The equipment configuration and SCOPE library tape should be the same as the one under which the overlay tape was prepared. Both I/O drivers and overlay information are assigned to high core; consequently, the overlay information may not fit into core if the I/O drivers needed for the equipment configuration exceed the space available. If an error occurs while LOADMAIN reads the MAIN program from tape, SCOPE writes a diagnostic on OUT and terminates the job.

**Example:**

<sup>7</sup><sub>9</sub>JOB, 7777, ROG, 50

<sup>7</sup><sub>9</sub>LOADMAIN, 25

data to be read by program

⋮

### 6.3

#### **LOADING OVERLAYS AND SEGMENTS**

SCOPE provides a subroutine to load overlays and segments (LOVER). A call may be generated to LOVER in COMPASS subprograms. LOVER is linked to the main subprogram when the overlay tape is prepared. FORTRAN source programs may call OVERLAY and SEGMENT to load and execute overlays and segments.

During execution of an overlay program, the main subprogram always remains in storage. Only one overlay and one segment may occupy storage at one time. The main subprogram may call overlays and segments from the overlay tape; overlays may only call their associated segments.

#### **RULES FOR LOADING**

1. Overlays and segments will normally be designed to be entered via the bank return jump instruction when LOVER is used, and to exit via their entry point, but they may exit directly to the calling overlay or main program.
2. An overlay may be loaded only by a call from the main program.
3. A segment may be loaded only by a call from the associated overlay if the overlay is in storage, or from the main program. A segment belonging to an overlay not currently in storage cannot be loaded.
4. Overlays can be called from the overlay tape in any order. Segments within overlays can be called from the overlay tape in any order.

These rules apply to FORTRAN and COMPASS programs.

**FORTTRAN CALL**    FORTRAN source language subprograms use the following call statement to load and execute overlays and segments. The overlay or segment call uses LOVER to load overlays and segments.

$$\text{CALL} \quad \left\{ \begin{array}{l} \text{SEGMENT} \\ \text{OVERLAY} \end{array} \right\} (o,s,u,d,p_1, \dots ,p_n)$$

SEGMENT    loads and executes a segment.

OVERLAY    loads and executes an overlay.

o            is the overlay number, specified for both segment and overlay.

s            is the segment number, blank or zero for an overlay.

u            is the logical unit number.

d            is a dummy parameter which must be present if any actual parameters appear. The dummy parameter may be blank.

p<sub>i</sub>          are actual parameters to be passed to the overlay or segment routine. No more than 59 may appear.

If o, s, u, or d is blank, the comma must appear; the order is fixed.

One subprogram in each overlay and segment must begin with the FORTRAN statement PROGRAM name. This statement may contain a maximum of 59 parameters:

PROGRAM name (p<sub>1</sub>, . . . ,p<sub>n</sub>)

name            is the transfer address for the overlay or segment.

p<sub>1</sub>, . . . ,p<sub>n</sub>    are formal parameters. The actual parameters, p<sub>i</sub>, in the CALL must correspond to these formal parameters.

**CALLING SEQUENCE**

The following calling sequence is generated during compilation for the CALL statement:

```

      BRTJ  ($)  { SEGMENT {
                OVERLAY } , , *
+   SLJ    *+m
      0n    DICT.
+   00     o
      00     s

```

```

+      00      u
        00      d
+      00      p1
        ⋮        ⋮
        00      pn

```

$$m = \frac{n+1}{2} + 1$$

n is the number of parameters specified in the FORTRAN CALL statement (o . . . p<sub>n</sub>).

The above calling sequence jumps to the OVERLAY or SEGMENT subroutine which passes the parameters, o, s, and u to LOVER. LOVER loads the segment or overlay and returns either a loading error code or the transfer address for the overlay or segment loaded.

If no errors occur during loading, the following call to the transfer address is generated by the SEGMENT or OVERLAY subroutine:

```

          BRTJ  ($)name, , *
+       SLJ   *+m
          0n   DICT.
+       00    p1
          ⋮    ⋮
          00    pn
name     is the transfer address for the loaded overlay or
          segment

```

$$m = \frac{n+1}{2} + 1$$

n is the number of actual parameters in the FORTRAN CALL statement, (p<sub>1</sub>, . . . , p<sub>n</sub>).

p<sub>1</sub>, . . . , p<sub>n</sub> are the actual parameters in the FORTRAN CALL statement.

**Example:**

CALL SEGMENT(3,2,25,,A,B,C)

The first FORTRAN card is: PROGRAM SUB2(X,Y,Z)

The transfer address in segment 2 of overlay 3 is SUB2.

The call to load the segment is:

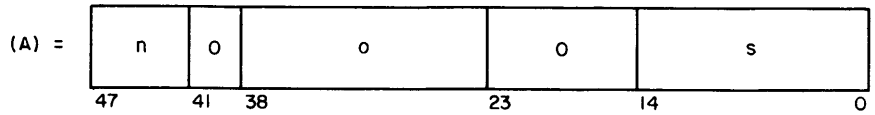
BRTJ	(\$)SEGMENT, , *
SLJ	*+5
07	DICT.
00	=03
00	=02
00	=025
00	0
00	A
00	B
00	C
00	0

The call from SEGMENT to the transfer address SUB2 in segment 2 of overlay 3 is:

BRTJ	(\$)SUB2, , *
SLJ	*+3
03	DICT.
00	A
00	B
00	C
00	0

**ERRORS** If errors occur during the loading of the overlays and segments, when using the FORTRAN CALL, the job is terminated. The A register will contain the contents of the parameters for the last LOVER call specified in the

CALL { OVERLAY }  
          { SEGMENT } statement.



n = Logical unit number  
o = Overlay number  
s = Segment number

The contents of the A register is written on OUT together with one of the following messages:

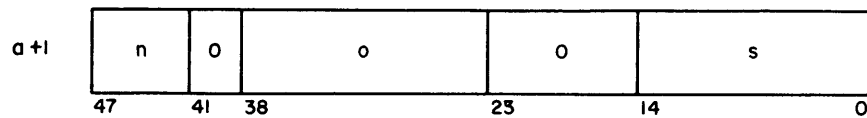
1. READ PARITY ERROR
2. LUN OUT OF RANGE
3. USE OF TOO MANY LUN
4. RECORD NOT ON THIS LUN
5. ILLEGAL SEQUENCE
6. OUT OF BOUNDS LOAD

**COMPASS CALL** The LOVER subroutine, which keeps a record of the last overlay and segment loaded from each unit, must be in storage as a part of the main subprogram. LOVER must be declared an external symbol to the main program in order for the SCOPE loader to load LOVER from the library tape and link it to the main program at the time the overlay tape is prepared. LOVER is called during program execution when an overlay or segment record is to be loaded from the overlay tape. LOVER does not execute the overlay or segment; it only calls it.

During execution of the overlay program, any residual tape beyond the double end-of-file marking the end of the overlay tape may be put to other uses. Therefore, when LOVER is entered, the overlay tape must not be positioned at the end of recorded information.

**CALLING SEQUENCE** The following calling sequence is included in a COMPASS subprogram to load an overlay or segment:

a CALL LOVER



a+2 (return point)



n = Logical unit number of the overlay tape  
o = Overlay number  
s = Segment number, S = 0 for the overlay

If LOVER has loaded the overlay or segment correctly, the A register is set to zero; a bank return jump instruction to the transfer address of the overlay or segment is placed in the Q register; return is made to the return point. The transfer address is stored in the second word of the overlay or segment record on the overlay tape.

**Example:**

The following macro definition could be used to specify calls to LOVER:

```
LOADOV    MACRO    (N,V,S)
           BRTJ    ($)LOVER,0,$LOVER
           VFD     A6/N, O3/0, A15/V, O9/0, A15/S
           EXT     LOVER
           ENDM
```

To load overlay 2 from logical unit 10, the following macro call can be made:

```
LOADOV    (10,2)
```

To load segment 2 of overlay 5 from logical unit 6, the following macro call can be made:

```
LOADOV    (6,5,2)
```

**ERRORS** If errors are encountered while loading the overlay or segment, the A register contains one of the following error codes right justified.

Error Code

- 1        A non-recoverable parity error was encountered when loading the overlay or segment record.
- 2        The specified logical unit number is not 1-49.
- 3        More than four logical units have been addressed in reading overlay and segment records.
- 4        The overlay or segment specified in the calling sequence was not on the specified logical unit.

- 5 The overlay or segment specified in the calling sequence was not consistent with the last overlay or segment loaded. (Either rule 3 or 4 in Section 6.3 has been violated.)
- 6 An attempt was made to load an overlay or segment out of bounds.

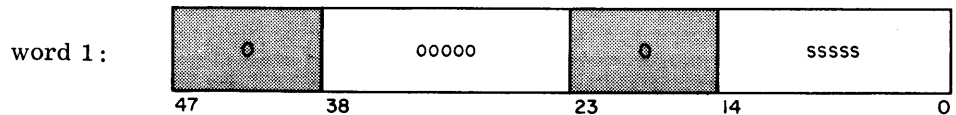
**6.4  
FORMAT OF  
OVERLAY TAPES**

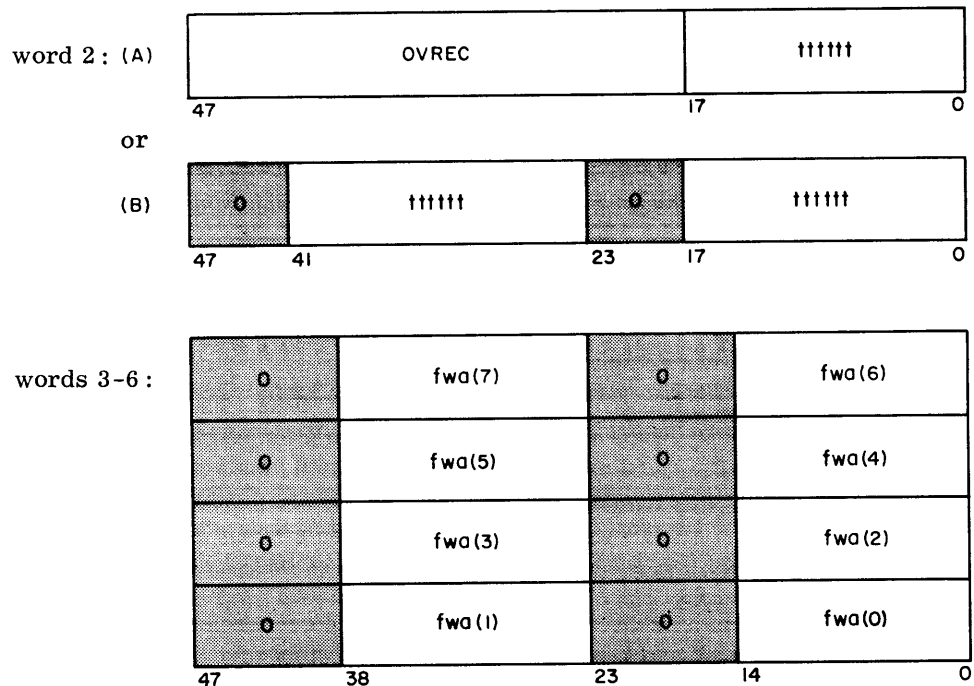
Each overlay tape is divided into files. The first file contains either one record, consisting of the main program, or no records. The second and remaining files are overlay files, the last of which is terminated with two end-of-file marks. Each file contains an overlay record, and records for associated segments. A single end-of-file mark precedes each overlay record. All the overlays and segments for one program must be on no more than four logical units, and no unit may occupy more than one reel.

The first record in each overlay file must be an overlay record and may be followed by segment records referenced by that overlay. The overlays on each overlay tape must be numbered in ascending order. Segments within an overlay file must be numbered and ordered consecutively. All segments of one overlay must be in one overlay file on one reel of one logical unit.

Loading information is contained in each overlay and segment record. The binary information in a record need not be all for one bank. A particular main, overlay, or segment may occupy any number of banks. In order to allow for the reloading of the various parts of a main, overlay or segment into the correct bank, the loader inserts 3600 control words in the record. The control word for loading a part of any overlay, segment or main program, n words in length, into a particular bank will be inserted within the record, immediately preceding the n words to be loaded. The n words will be followed by another control word for loading the next m words into another bank, and so on, until the last two words of the record are the control words, IOTR 0,0.

The first six words of each record contain identification parameters for that overlay or segment. These six words are in the following format:





ooooo is the overlay number  
 sssss is the segment number  
 OVREC is the BCD mnemonic  
 ttttt is an 18-bit transfer address  
 fwa(n) is the first word address of available storage in bank n at the time the record is written.

If the record is a main record, word 1 is zero. If there are two transfer addresses for the main record, word 2 type (B) is used. The first transfer address encountered by the loader occupies bits 41-24; the second transfer address occupies bits 17-0. If only one transfer address is encountered, it occupies bits 17-0, and is in the format for type (A) of word 2.

If bits 14-0 of word 1 are zero, the record is an overlay record.

All overlay and segment records contain the mnemonic OVREC in bits 47-18 of word 2.

Words 3-6 of each record contain 15-bit addresses in the upper and lower address portions of each word. Each address specifies the last word address+1 of any numbered common in a particular bank at the time the record was written. For a non-existent bank the address contains zero. The last word address+1 of numbered common is the same as the first word address of available storage for a bank.

A typical overlay record might be composed as follows:

000	00006	000	00003	word 1
OVREC			123456	word 2
000	00000	000	00000	words 3-6
000	00000	000	00000	
000	00100	000	00001	
000	01234	000	17654	
IOTW,C		(1)	10000,100	
{ 100 words absolute binary subprogram }				
IOTW,C		(2)	70000,300	
{ 300 words absolute binary subprogram }				
IOTW,C		(3)	60000,200	
{ 200 words absolute binary subprogram }				
IOTR		(0)	0,0	
IOTR		(0)	0,0	

This example is segment 3 of overlay number 6. The lower limits of available storage for this segment are: (0)17654, (1)01234, (2)00001, (3)00100. The record specifies loading 100<sub>g</sub> words into bank 1 beginning at address 10000, 300<sub>g</sub> words in bank 2 beginning at address 70000, and 200<sub>g</sub> words into bank 3 beginning at address 60000. The transfer address for this segment is 23456 in bank 1.

# LIBRARY PREPARATION AND MAINTENANCE

# 7

---

PRELIB is the SCOPE routine which prepares and maintains library tapes. A library tape may include:

- Absolute binary programs
- Relocatable binary programs
- Binary data
- BCD data
- Subroutine directories
- User defined information
- End of file marks

New libraries may be created by extracting records from up to nine old libraries, and/or processing data in any one of five modes from one or more input units. The old libraries may be the standard SCOPE library (logical unit 70) or the auxiliary libraries (logical units 72-79). The input units may be the standard input unit (logical unit 60) or a programmer defined unit (logical units 1-49, or 69). The five modes for processing input records are absolute binary, relocatable binary, BCD data, binary data, or special.

Directories describe the relocatable binary records which follow them on the library tape. The loader uses the information contained in the directory to locate and load subroutines.

The first record of every library contains a table of contents and a label. The table of contents, which describes each record on the tape, is used by SCOPE to position the library tape and to locate directories. Although auxiliary libraries are referred to by logical unit numbers 72-79 during PRELIB processing, all library tapes contain logical unit number 70 in the label; however, the labels may contain different names and edition numbers.

An arrangement of an auxiliary library tape could include a directory for relocatable routines followed by an end-of-file card, a second DIRECTORY for subsequent relocatable routines and an end and end-of-file card and so on. The SCOPE system routines need not be included on auxiliary library units. When they are included, they may be called by the programmer but not by SCOPE. A complete auxiliary library contains at least one directory including all REL records and any others desired. If the directory is the first record on the tape, the table of contents for that tape will be part of the first record.

The PRELIB routine may be used to:

List the table of contents of a library

Edit an existing library while inserting records from an input unit

Prepare a new library from the old SCOPE library, auxiliary libraries, and records from an input unit

PRELIB processing consists of two phases. In phase 1, control cards are read from the standard input unit and records are processed according to the mode specified. Those records that can be fully processed before reading another control card (absolute, relocatable, binary data, BCD data, special) are written on the scratch unit, S0, as soon as they have been processed. Directories, initiated by DIR control cards and terminated by END control cards, are prepared and stored internally. A table of contents is developed internally containing the name of each record and its mode.

When PRELIB encounters the FINISH control card, the run will be terminated if it was a LIST run. If it was EDIT, the rest of the tape is copied onto S0. If it is either an EDIT or PREPARE run, S0 is rewound and the table of contents of the new library is listed on OUT. If any errors are found during phase 1 processing, the run is terminated. If no errors occur, phase 2 is initiated. The new library tape will be created on logical unit 71, the label will contain logical unit 70 and the name for logical unit 71. The first record on S0 must be binary; the new table of contents plus the label is appended to the first record and written on the new library.

According to their order and mode in the table of contents in storage, records are copied from S0 onto 71, directories are copied from storage onto 71 and end-of-file marks are written on 71 where specified. If a record on S0 has been defined in phase 1 as repeated, when it is encountered on S0, it is copied into a repeat table in storage. If it is too long for storage it is written on S1. When the repeated record is called for in the table of contents, it is written onto logical unit 71. Upon completion of the new library tape, 71 is rewound and unloaded and control is returned to SCOPE. An end-of-file mark is not written at the end of the new library unless specified.

Only parity errors are possible during phase 2. When an unrecoverable parity error occurs, the message PARITY ERROR will be written on OUT after the table of contents listing; no further records will be written on 71.

## 7.1 INPUT/OUTPUT

All control cards to a PRELIB run are read from the standard input unit (60). Column 1 of PRELIB control cards must contain a 7,9 punch; the balance is free-field, except for imbedded blanks, which are significant. A control statement must be contained on one card and is terminated by a period or column 80. Comments, which may appear beyond the period, are ignored by PRELIB.

Input to PRELIB may be records from logical units 1-49, 60, and 62, from the SCOPE library (logical unit 70), and from auxiliary libraries (logical units 72-79).

Input records in absolute, relocatable, BCD data, binary data, and special form must be indicated by the control statements, ABS, REL, BCD, BIN, special. The records may follow the control statements directly on logical unit 60. Records contained on a different logical unit (1-49, or 62), must be specified by a UNIT control statement. UNIT may specify 60; if no UNIT statement is used, 60 is assumed.

During a PRELIB run, the following information is written on the standard output unit:

- all PRELIB control cards
- the new table of contents
- error messages
- MAP of absolute and relative records

## 7.2

### RECORD TYPES

The mode of transfer for records from logical units 1-49, 69 to the new library on logical unit 71 must be specified with a REL, ABS, BIN, or BCD control card. Other control statements define directories and end-of-file marks on the new library tape.

In the following statements, name is entered in the table of contents to identify the information following the control statement on the logical unit. It may be up to 31 characters in length. If no name is specified, the name of the first subprogram in the record is inserted in the table of contents; if no subprogram name exists, blanks are inserted. For relocatable binary records, the name from the IDC card is inserted in the directory which contains the record.

### RELOCATABLE BINARY RECORDS

$\begin{matrix} 7 \\ 9 \end{matrix}$ REL,name

REL specifies that the information on a logical unit is a single subroutine in relocatable binary card form and should be written as one record on the new library in a condensed relocatable form. REL may be used only within the range of a directory. A DIR control statement must precede the first REL control statement.

REL processes the single relocatable binary subroutine by placing the information from the IDC, EPT, BLT, and EXT cards in the subroutine directory tables. BRT, RBD, LAT, and TRA cards are read into storage in card image format. When one TRA card is processed, the entire record is written on S0; another control statement is read from the standard input unit.

## ABSOLUTE BINARY RECORDS

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ ABS,name(location)

PRELIB links relocatable binary subprograms read from the standard input unit or another logical unit and writes them on S0 as one absolute binary record. An absolute binary record may originate at any absolute location specified in the location field (in octal). If the location is not specified, the absolute binary record originates at location zero.

ABS processes the relocatable binary subprograms by loading the RBD card images as though the first subprogram began at the specified absolute location and subsequent subprograms began where the preceding subprogram terminated. The subprograms are linked by associating external symbols with entry points within the program. When two consecutive TRA cards are encountered, the linked subprograms are written as one record on S0.

### **Examples:**

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ ABS,BOOT

This record, BOOT, is originated at absolute location zero.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ ABS,LDR(14000)

This record, LDR, is originated at absolute location 14000.

## BINARY RECORDS

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ BIN,name

Binary information is transferred without alteration from the standard input unit or another logical unit to S0. It is written on S0 as a single binary record. The end of the data is indicated by a card with a word count of zero.

## BCD RECORDS

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ BCD,name

BCD information is transferred without alteration from the standard input unit or another logical unit to S0. It is written on S0 as a single binary record in internal BCD. The end of the data is indicated by a blank card.

The data appears on the library tape as a binary record in internal BCD.



**MACRO  
DEFINITIONS**

<sup>7</sup>MACRO  
<sub>9</sub>

MACRO composes the system and library macro tables for COMPASS. Macro definitions in COMPASS language are read from the standard input unit or another logical unit. The first set of definitions is the system macro table and is terminated by ENDSYS\* following the last ENDM. Immediately following the ENDSYS\* card are the library macro definitions. The table is terminated by an ENDLIB\*. If there are no library macro definitions, the ENDSYS\* card need not appear.

When an ENDLIB\* card is encountered, the entire table is written as one binary record on S0 and control is returned to PRELIB.

**Example:**

```
7MACRO  
9  
NAME1      MACRO      (p1,p2, . . . , pn)  
      .  
      .  
      ENDM  
NAME2      MACRO      (p1,p2, . . . , pn)  
      .  
      .  
      ENDM  
      ENDSYS*  
NAME3      MACRO      (p1,p2, . . . , pn)  
      .  
      .  
      ENDM  
      ENDLIB*
```

**USER CONTROL  
CARD**

The user may add special routines to PRELIB. The special control statement, 7 characters or less, must be added to the PRELIB list of control statements. Input/output will be handled by PRELIB if jumps are made to the proper routines with the correct parameters. If a special routine results in one or more records on 71, the names and modes of the records must be added to the table of contents.

## DIRECTORIES

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ DIR,name

The DIR control statement indicates that a directory is to be inserted on the new library tape, 71. A directory of all subsequent relocatable subprograms is constructed in storage until an END, DIR, or FINISH control statement is encountered. Only relocatable binary subprograms may be contained in a directory.

The directory name and mode are entered in the table of contents. The directory is constructed in storage from tables or, during the processing of IDC, EPT, BLT, and EXT cards, in the relocatable binary subprograms. During phase 2 when a directory mode is encountered in the table of contents, the directory is written on the new library tape, logical unit 71.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ END,name

This control statement terminates the relocatable binary subprograms to be included in the directory specified by the previous DIR control statement. END does not result in a record on the new library tape.

### **Example:**

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ DIR,name

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ REL,name

relocatable binary subprogram

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ REL,name

relocatable binary subprogram

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ END,name

## END-OF-FILE

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ EOF,name

EOF specifies that an end-of-file mark is to be written on the new library, logical unit 71. When the EOF control statement is read from the standard input unit, the name and mode are entered in the table of contents in storage. During phase 2 when an end-of-file mode is encountered in the table of contents, an end-of-file mark is written on the new library tape.

End-of-file marks placed before each directory, allow SCOPE to skip files to locate directories. If a loader precedes a directory, the end-of-file should precede the loader. End-of-file marks should not be used within the directory.

It is recommended that a library be terminated by two consecutive end-of-file marks.

### 7.3 CONTROL STATEMENTS

Control statements on the standard input unit may prepare a new library tape, edit an old library tape or list a library table of contents. Records can be repeated and different logical units may be specified to contain input records. PREPARE, EDIT, and LIST are mutually exclusive within a PRELIB job.

The parameter, libname-ee, in a control statement refers to the name in the label of a library tape. The tape edition number, ee (1 to 99), is optional. For the SCOPE system library tape, logical unit 70, the name is \*. LIB is never used.

A libname beginning with an \* means the current system library unit 70. Any characters immediately following the \* and before the next comma or right parenthesis are ignored. PRELIB deletes leading blanks and fills in trailing blanks so that all names are 32 characters in length, imbedded blanks are counted.

The parameter, rec, in a control statement refers to an existing library record which is to be transferred to the new library. When rec is numeric, it specifies the ordinal of the record on the source library which is the same as in the source library table of contents. When rec is alphabetic, it specifies the name of a record listed in the table of contents of the source library.

A range of record identifiers may be specified inclusively as rec<sub>j</sub>-rec<sub>k</sub>.

### PRELIB RUN

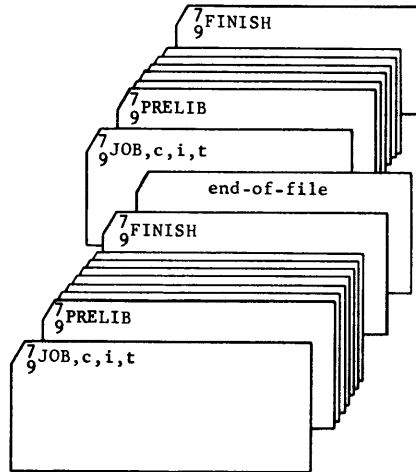
A PRELIB run may be PREPARE, EDIT or LIST; it begins with a PRELIB control statement and terminates with a FINISH control statement. There may be only one PRELIB run in a SCOPE job.

#### <sup>7</sup>PRELIB <sub>9</sub>

Calls the SCOPE library preparation routine into storage and transfers control to it.

#### <sup>7</sup>FINISH <sub>9</sub>

Returns control to SCOPE after processing is completed.



**COMMENTS**

A comment card is ignored by PRELIB and printed on the standard output unit.

7\* . . .  
9

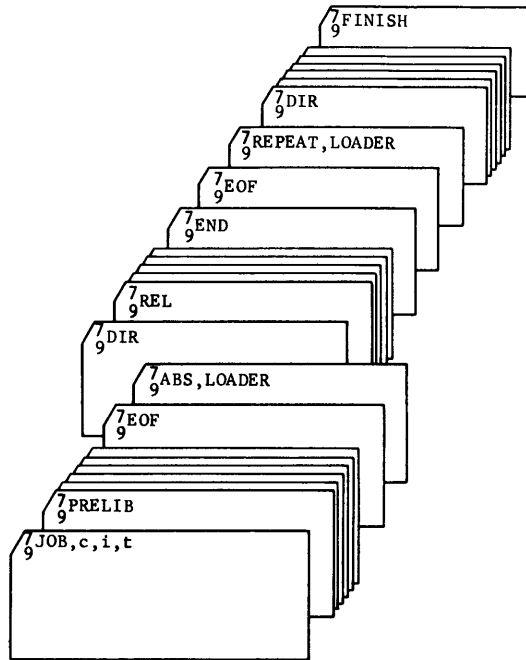
In a comment card, the first non-blank character must be \*.

**REPEATED RECORDS**

The same record may be written in different positions on the new library by a REPEAT statement.

7  
9 REPEAT,name

name may be any record that has already been processed except an EOF or DIR. To repeat a REL record, it must appear in a completed directory. Two records with the same name may not be in the same directory.

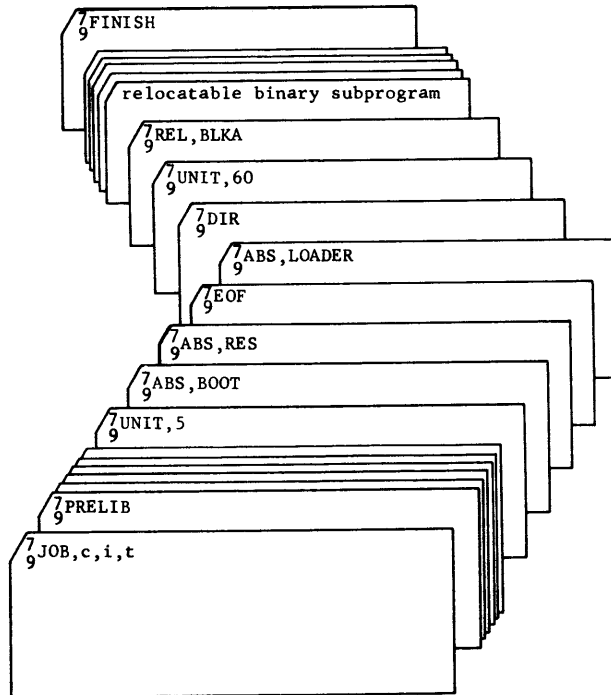


**LOGICAL UNIT  
FOR INPUT  
RECORDS**

A UNIT control statement specifies the logical unit, 1-49, or 69, from which input records will be read until the next UNIT control statement is encountered on INP or until the end of the PRELIB run.

$\begin{matrix} 7 \\ 9 \end{matrix}$  UNIT,u

A control statement specifying mode, such as REL or ABS, is read from INP before the input record is read from the specified logical unit.



During processing, PRELIB encounters the UNIT,5 control statement.

- 1) The UNIT,5 control statement indicates that input records are on logical unit 5.
- 2) ABS,BOOT specifies that PRELIB is to read the next input record, BOOT, in relocatable binary from logical unit 5 until 2 consecutive TRA cards are encountered. BOOT is written on S0 in absolute binary.
- 3) ABS,RES specifies that RES is to be read from unit 5 and written on S0 in absolute binary.
- 4) An end-of-file is indicated in the table of contents in storage.
- 5) ABS,LOADER specifies that the LOADER is to be read from unit 5 and written on S0 in absolute binary.
- 6) A directory for the following relocatable binary subprograms will be built in storage.

- 7) UNIT,60 indicates that input records are on the standard input, logical unit 60.
- 8) REL,BLKA specifies that PRELIB is to read the next input record, BLKA, a relocatable binary subprogram, from the standard input unit until 1 TRA is encountered. BLKA is written on S0 in relocatable binary.

## LISTING TABLE OF CONTENTS

The table of contents of the named library tape is written on the standard output unit. As many as 9 tapes may be listed in one run if one is \*, the SCOPE system tape.

<sup>7</sup><sub>9</sub>LIST(libname<sub>1</sub>-ee,libname<sub>2</sub>-ee, . . .)

PRELIB considers the parameters as logical units 72-79 from left to right in the statement. If a libname cannot be found, a message is typed on OUT. (Messages and Diagnostics, Appn C.) If more than one tape with the same name is specified, the edition numbers of all but the last in the list must be specified. If this is not done, assignments must be made on the typewriter.

### Examples:

<sup>7</sup><sub>9</sub>LIST(\*,SASY1-8,BRT SCOPE,MACRO,SASY1)

PRELIB considers libnames to be logical unit numbers:

*	70
SASY1-8	72
BRT SCOPE	73
MACRO	74
SASY1	75

For logical unit number 75, PRELIB will choose the library named SASY1 which has not yet been assigned a logical unit number.

If there is no mounted tape with BRT SCOPE in the label (including the space) a message will be typed: 73 NEEDED

<sup>7</sup><sub>9</sub>JOB,777,DDSTON,3

<sup>7</sup><sub>9</sub>PRELIB

<sup>7</sup><sub>9</sub>LIST(SASY1-1,SASY1-2,\*)

<sup>7</sup><sub>9</sub>FINISH

The table of contents for SASY1-1, SASY1-2, and the SCOPE system library are listed on the standard output unit.

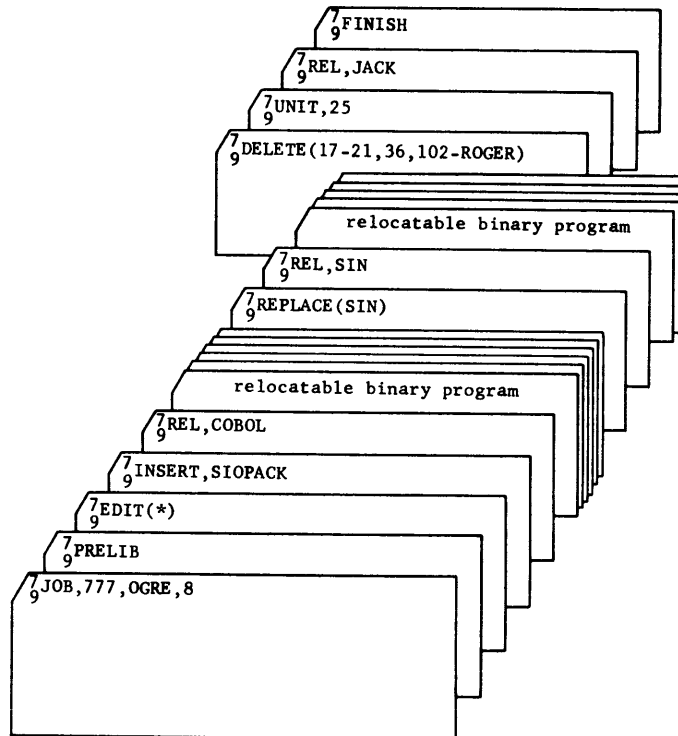
## EDITING EXISTING LIBRARY

A single library tape is edited by deleting, replacing or inserting records. Auxiliary libraries may not be used.

**EDIT** is the first statement in the editing deck.

```
7 EDIT(libname-ee)  
9
```

Edit Example





The source library tape is the SCOPE system library.

- 1) The source library is copied onto S0 up to and including SIOPACK and the tape is positioned at the beginning of the record following SIOPACK.
- 2) The relocatable program, COBOL, is copied from the standard input unit onto S0.
- 3) The source library is copied beginning with the record following SIOPACK, skipping SIN, and the tape is positioned at the beginning of the record following SIN.
- 4) The relocatable program, SIN, is copied from the standard input unit onto S0.
- 5) The source library is copied onto S0 beginning with the record following SIN through record 101, skipping records 17-21 and 36. Records 102 through ROGER are skipped and the source library is left at the beginning of the record following the one named ROGER (assuming that ROGER comes after 102).
- 6) The relocatable binary subprogram, JACK, is then copied from logical unit 25 onto S0.
- 7) FINISH indicates that the remainder of the source library is to be copied onto S0. The new table of contents is listed on the standard output unit. The new library is then copied from S0 and storage onto 71. Note that if a directory is within the range of records to be copied, that the directory is not copied, but that a completely new directory is built in storage and written on 71 at the location specified.

The name of the new library will be the same as the source library with the edition number incremented by one. An \* is used if the SCOPE system library is to be edited.

All records on the source library, except those replaced or deleted, will be copied onto S0. Since the source library can move forward only, records must be declared in the order in which they appear on the source tape.

During editing, old directories are updated by PRELIB.

The FINISH control statement terminates the editing deck; after it is encountered, the rest of the source library is copied onto S0. The new table of contents is listed on OUT; then the new library is written.

**Example:**

```
7  
9EDIT(LIBRARY-5)
```

The tape to be edited is named LIBRARY, edition number 5. The new tape will be named LIBRARY, edition number 6.

**DELETE** specifies records,  $rec_1$ , on the source library which are not to be copied; all other records not to be deleted are copied.

```
7  
9DELETE(rec1,rec2,rec3-rec4, . . . ,recn)
```

DELETE may be followed by control statements to insert any number of records onto the new library to effect a REPLACE control statement.

Although an END control statement is not a record, it is indicated in the source table of contents. To delete it from the source table of contents, the record following it must be deleted.

**Example:**

```
7  
9DELETE(1-4,COMPASS,17-FTN,PRELIB)
```

```
7  
9REL,SIN
```

relocatable binary subprogram

.  
. .  
.

While copying the source library on to S0:

- 1) Records 1-4 are skipped.
- 2) Records 5 to COMPASS minus 1 are copied onto S0.
- 3) COMPASS is deleted.
- 4) COMPASS plus 1 to record 16 are copied onto S0.
- 5) 17 to FTN are skipped.
- 6) FTN plus 1 to PRELIB minus 1 are copied onto S0.
- 7) PRELIB is skipped.

- 8) The source tape is positioned at the beginning of PRELIB plus 1.
- 9) Relocatable binary subprogram SIN is read from the standard input unit, processed, and written on S0.

**INSERT** causes all records up to and including a specified record to be copied onto S0. The mode of the information from INP is indicated by REL, ABS, BIN, BCD, DIR, END, EOF and special control cards. Records will be copied from INP until another INSERT, DELETE, REPLACE or FINISH control card is encountered. The source tape is positioned after the specified record. INSERT is followed by control statements to insert any number of records on the new library at that point.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ INSERT, rec<sub>i</sub>

If rec<sub>i</sub> is 0, any input records following INSERT will be copied before any records are copied from the source library.

**Example:**

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ ABS,BOOT

relocatable binary program

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ INSERT,ALGOL

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ REL,SORT

relocatable binary subprogram

.  
.  
.

- 1) BOOT is written in absolute binary on S0.
- 2) The source library is copied onto S0 up to and including ALGOL and is positioned at the beginning of ALGOL plus 1.
- 3) The relocatable binary subprogram, SORT, is copied onto S0.

**REPLACE** Removes specified records from the source library as it is copied onto S0. The source library tape is positioned after the specified records. REPLACE may be followed by control statements to insert any number of new records at that point. There need not be a one-to-one correspondence between the number of records removed and inserted.

```
79REPLACE(rec1, rec2, rec3-rec4, . . . , recn)
```

**Example:**

```
79REPLACE(A-14)
```

```
79UNIT,7
```

```
79REL,COMPASS
```

```
.  
. .  
. .
```

- 1) The source library is copied onto S0 skipping records A-14. The source library is positioned at the beginning of record 15.
- 2) The relocatable binary subprogram, COMPASS, is copied from logical unit 7 to S0.

**NEW LIBRARY  
TAPE**

A new library is prepared by extracting information from existing libraries, and including new records from a logical unit. When the FINISH statement is encountered, information is taken from S0 and storage, the new table of contents is written on OUT, and the new library is written on logical unit 71. During library preparation, records need not be declared in the order of appearance on the library tapes. If a record is requested that has been passed, the library tape will be rewound, then spaced forward to extract the requested record.

**PREPARE** instructs PRELIB to prepare a new library from the source libraries.

```
79PREPARE, name1 (name2, name3, . . .)
```

Name1 designates the new library tape (logical unit 71). Name2, name3, and so on, specify library tapes (logical units 72, 73, . . .) from which information will be taken to form the new library. A name beginning with an asterisk specifies the current library (unit 70). Any characters between the \* and the next

comma or right parenthesis are ignored. PRELIB deletes leading blanks and fills with trailing blanks so that all names are 14 characters in length (embedded blanks are meaningful). If there is not enough room on one PREPARE card for all of the names in the parameter list, extra PREPARE cards may be used. Name1 on subsequent cards will be ignored, but the field must be indicated by a comma if the name is omitted, for example, PREPARE, (name4, . . .).

**EXTRACT** specifies the records on source library tapes which are to be transferred to the new tape. EXTRACT can only be used with a PREPARE statement.

```
79EXTRACT, u(rec1, rec2, . . . , recn)
```

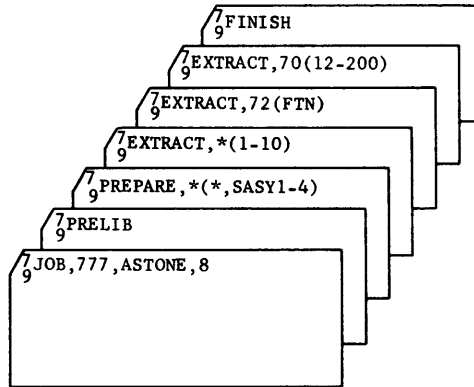
u is the logical unit number (\*, 70, 72-79). Either \* or 70 may be used for the current system library. A record identifier may be a three-digit number specifying the ordinal of the record on a library tape or it may be alphabetic specifying the name of a record which is found in the table of contents of u. For example, an alphabetic record identifier may be a COMPASS alphanumeric symbol or a COBOL name.

The records to be extracted from the specified library tape, rec<sub>i</sub>, need not be specified in the order of appearance on the library tape, since the library tape can be rewound. If the same name appears in the table of contents twice, the first will be chosen.

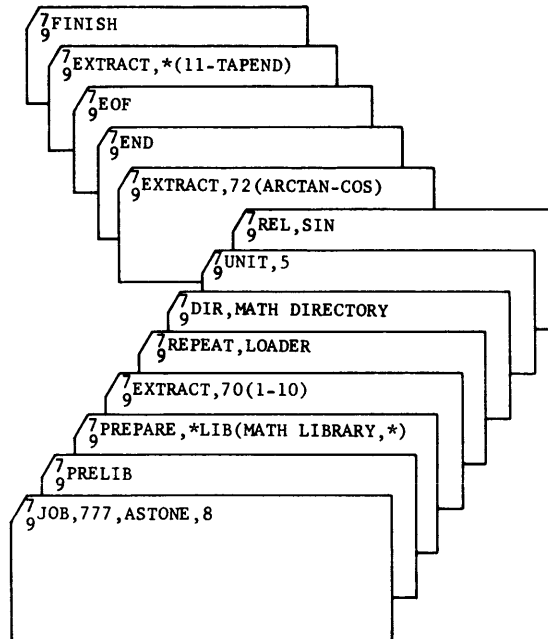
**Example:**

```
79EXTRACT, *(1-5,4,19-FTN,12)
```

Records 1-5 are copied onto S0 from the system library; the library tape is rewound. Records 1-3 are skipped, record 4 is copied, records 5-18 are skipped, records 19-FTN are copied. The tape is rewound and records 1-11 are skipped then record 12 is copied. The library is positioned at the beginning of record 13 at this point. If FTN precedes record 19 on the tape, that tape will be rewound after copying 19, then skipped forward to copy FTN.



- 1) A new SCOPE system library is prepared from the current library and SASY1-4.
- 2) Records 1-10 are copied from \* to S0.
- 3) FTN is copied from logical unit 72 to S0.
- 4) Records 12-200 are copied from logical unit 70 to S0.

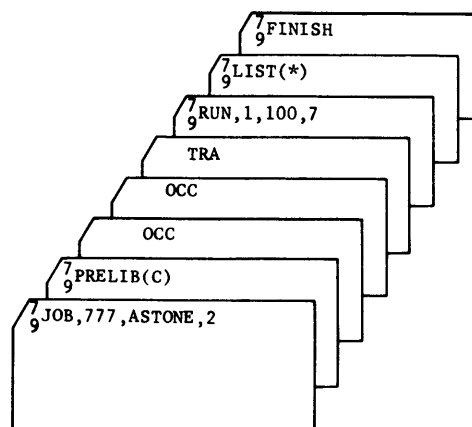


- 1) A new SCOPE system library will be prepared from the current system library and the math library.
- 2) The first 10 records are copied from the current system library onto S0.
- 3) The loader must have already been encountered on the system library; indication will be made to repeat it.
- 4) A math directory will be built in storage to include the following relocatable binary records.
- 5) The relocatable binary input record, SIN, will be read from logical unit 5, processed, and written on S0.
- 6) ARCTAN-COS is copied onto S0 from the math library.
- 7) The math directory is terminated by the END control statement.
- 8) EOF is written in the new library table of contents.
- 9) Records 11-TAPEND are copied from the current system library to S0. Naming the last EOF in a library provides a convenient way to read to the end of the library.
- 10) Library preparation is terminated with FINISH.

#### 7.4

**CHANGING PRELIB** The SCOPE parameters,  $s_1$  and  $s_2$ , may be used to temporarily modify, snap, or trace PRELIB. The program extension area may not be used with octal corrections.

$\begin{smallmatrix} 7 \\ 9 \end{smallmatrix}$ PRELIB( $s_1, s_2$ )



## 7.5

**TABLE OF  
CONTENTS**

The library table of contents is in the first record on a library tape between the bootstrap routine and the tape label. Records are written in binary and numbered in the order in which they appear on the library.

Entries are in the order specified by the control statements used to prepare the library. An entry may or may not describe a record in the library. If it does, the mode, name, and record number are recorded. If the entry does not describe a record, such as END which is not a record on the library, mode and name are recorded. If the name is longer than one word, the first character gives the number of words in the name.

The table of contents of each new library is listed on the standard output unit during library preparation or editing. The LIST control statement may specify that the table of contents of an existing library is to be listed.

Following is a sample SCOPE system library, SASY1, as listed on the standard output unit.

```
LABEL 5( )70000SASY1          0104226406
```

<u>Record Number</u>	<u>Name</u>	<u>Mode</u>
1	BOOT	ABS
2	RESIDENT	ABS
3		EOF
4		DIR
5	CARPU	REL
6	DR3649	REL
		END
7		EOF
8	LOADER	ABS
9		DIR
10	SIOPACK	REL
11	COMPASS	REL
12	COMPASSX	REL
13	FTN	REL
		END
14		EOF
15	LOADER	ABS
16		DIR



<u>Record Number</u>	<u>Name</u>	<u>Mode</u>
17	IOP	REL
.	.	.
.	.	.
.	.	.
.	.	.
87	DPOWER	REL
		END
88		EOF
89		DIR
90	DCP	REL
		END
91		EOF
92		DIR
93	STD	REL
94	PRELIB	REL
		END
95		EOF
96		EOF
97	TAPEND	EOF
	END OF LIBRARY TAPE	

## **APPENDIX SECTION**

# AVAILABLE EQUIPMENT TABLE

# A

This table is a record of the equipment at an installation. It may be altered temporarily in storage by the AET statement (Sec. 2.6). The format of each word in the table is as follows:

h	i/o	s	sd	x	cr	sc	a	e	u	d	
6	2	1	6	1	8	6	2	3	6	7	
47	41	39 38		32 31		23	17	15	12	6	0

<u>h (octal)</u>	<u>Hardware Type</u>	<u>Mnemonic</u>
01	magnetic tape unit	MT
02-03	(reserved for tape-like equipment)	
04	card reader	CR
05	card punch	CP
06	line printer	LP
07	paper tape station	PT
10	typewriter	TY
11	disc file	DF
12	drum	DR
13	CRT display	TV
14	plotter	PL
15-17	---	
20	3682 Satellite	
21	equipment associated with Satellite 1 (SA)	
22	equipment associated with Satellite 2 (SB)	defined within the Satellite system
.	.	
.	.	
.	.	
26	equipment associated with Satellite 6 (SF)	
27-57	---	
60-77	reserved for installation use	

i/o input-output capability of the unit

01	output
10	input
11	input and output

s SCOPE accessibility bit

0	unit is accessible to SCOPE
1	unit is accessible only to Satellites.

sd ordinal of the driver for all Satellites capable of using the unit. This ordinal indicates the distinction between units of the same hardware type which are programmed differently such as 3641 and 3649 card readers.

x asynchronous interrupt capability of the unit.

1	unit is capable
0	unit is not capable

cr ordinal (>0) of the unit controller.

sc Satellite-control-channel field used as follows:

If  $h = 21-26$ ,  $sc =$  hardware type for the Satellite

If the equipment is capable of interrupting the 3600 asynchronously,  $sc =$  SCOPE channel connection and  $e =$  SCOPE equipment number

When a unit is connected,  $sc =$  channel connection. ( $sc = 40B$  initially)

a availability

00	unassigned, available
01	unassigned, down
10	assigned to SCOPE
11	assigned to Satellite

e equipment code of the unit for the Satellite.

u the unit code.

d the ordinal ( $d>0$ ) of the driver name.

# MACRO DEFINITIONS AND CALLING SEQUENCES

**B**

## INTERNAL

```

CALL36      MACRO      (CC)
+           63         CC*8
           03         SENTRY
           EXT        SENTRY
           ENDM
    
```

SCOPE call codes (CC) are as follows:

0	EXIT	17	REMOVE
1	READ	18	LIMIT
2	WRITE	19	FREE
3	REOT	20	TIME
4	WEOT	21	BOUND
5	BSPR	22	UNBOUND
6	BSPF	23	DATE
7	REWIND	24	LOADER
8	UNLOAD	25	LIBRARY
9	SKIP	26	MEMORY (CORE)
10	ERASE	27	HERESAQ
11	MARKEF	28	RELEASE
12	MODE	29	RDLABEL
13	STATUS	30	WRLABEL
14	LABEL	31-63	Not Assigned
15	SAVE (UNSAVE)	64-4095	Reserved for individual installation
16	SELECT		

```

IOF      MACRO      (L, C, R, I, CC)
          CALL36    (CC)
          control word address (C)
          logical unit number (L)
          STAR      ( (R), *-2)
          interrupt address (I)
          ENDM
    
```

```

STAR     MACRO      (R, S)
          IFT, EQ    R, /*, 1
          reject address (S)
          IFT, NE    R, /*, 1
          reject address (R)
          ENDM
    
```

```

Z          MACRO      (CC, I, X)
          VFD         A6/CC, A3/X, A15/I
          ENDM

```

## INPUT/OUTPUT

### READ, WRITE, REOT, WEOT

#### Definition

```

<READ>
<WRITE>
<REOT>
<WEOT>
MACRO      (L, C, R, I)
IOF        ( (L), (C), (R), (I), <1>
          <2>
          <3>
          <4> )
ENDM

```

#### Calling Sequence

```

<READ>
<WRITE>
<REOT>
<WEOT>
63         <1>
          <2> *8
          <3>
          <4>
03         SENTRY
00         control word address
00         logical unit number
00         reject address
00         interrupt address

```

NOTE: The names of these macros are used for illustrative purposes, and may be any acceptable alphanumeric identifier.

### RDLABEL, WRLABEL

#### Definition

```

<RDLABEL>
<WRLABEL>
MACRO      (L, B, R, I)
IOF        ( (LL), (B), (R), (I), <29>
          <30> )

```

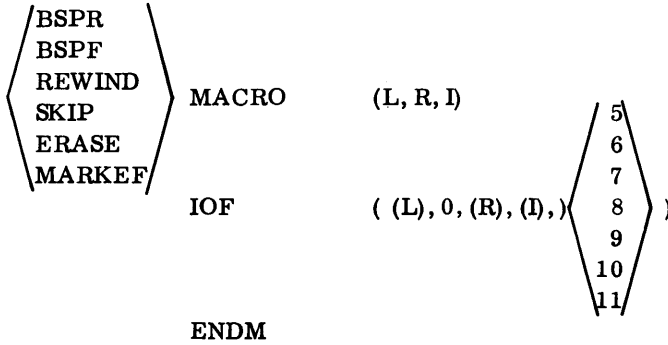
#### Calling Sequence

```

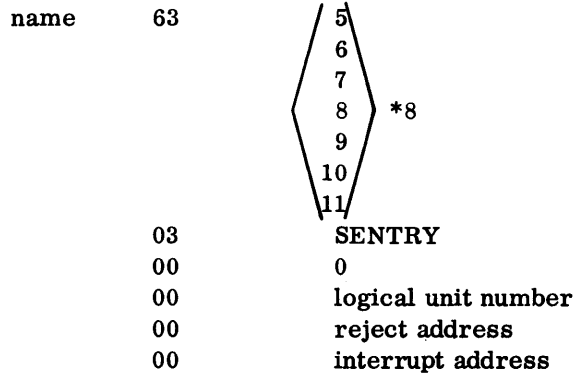
<RDLABEL>
<WRLABEL>
63         <29>
          <30> *8
03         SENTRY
00         buffer address (B)
00         logical unit number (L)
00         reject address (R)
00         interrupt address (I)

```

**BSPR, BSPF, REWIND, SKIP, ERASE, AND MARKEF**



**Calling Sequence**



The following mnemonics may be used to reference scratch and system units:

<u>Mnemonic</u>	<u>Logical Unit</u>	<u>Mnemonic</u>	<u>Logical Unit</u>
S0	50	INP	60
S1	51	OUT	61
S2	52	PUN	62
S3	53	ICM	63
S4	54	OCM	64
S5	55	ACC	65
S6	56	LGO	69
S7	57	LIB	70
S8	58	SCR	80
S9	59		

## UNLOAD

UNLOAD	MACRO	(L, R, I, P)
	IOF	( (L), (P), (R), (I), 8)
	ENDM	

### Calling Sequence

name	63	8*8	
	03	SENTRY	
	pp	release parameter	
	00	logical unit number	
	00	reject address	
	00	interrupt address	
pp =	0	logical unit assignment is to be released after the physical unit is unloaded.	
	≠	0	logical unit assignment is not to be released after the physical unit is unloaded.

## RELEASE

RELEASE	MACRO	(L, R, P)
	CALL36	(28)
	IFZ	release parameter (P), 1
k	EQU	0
	IFN	release parameter (P), 1
k	EQU	1
	IFT, EQ	reject address (R), /*/, 1
	VFD	A1/k, O5/0, A18/*-1
	IFT, NE	reject address (R), /*/, 1
	VFD	A1/k, O5/0, A18/reject address (R)
		logical unit number (L)
	ENDM	

### Calling Sequence

name	63	28*8
	03	SENTRY
	VFD	release parameter, reject address (R)
	00	logical unit number (L)



## MODE

MODE	MACRO	(L, R, U, F, D, DR)
	CALL36	(12)
	STAR	( (R), *-1)
		logical unit number
	VFD	O23/0, A5/U, A5/F, A5/D, A5, DR
	ENDM	

### Calling Sequence

name	63	12*8
	03	SENTRY
	00	reject address
	00	logical unit number
	VFD	O23/0, A5/U, A5/F, A5/D, A5, DR
(usage)	U {	RW 24B allow all legal operations
		BY 22B bypass unit
		RO 21B allow only input operations
(format)	F {	BCD 7 set BCD recording mode
		BIN 6 set binary recording mode
(density)	D {	HY 13B 800 bpi
		HI 12B 556 bpi
		LO 11B 200 bpi
		OP 10B operator's option
(direction)	DR {	RV 31B read reverse
		ND 30B read normal

## STATUS

STATUS	MACRO	(L, M)
	CALL36	(13)
	IFT, EQ	M, /M/, 1
	00	0
-	00	logical unit number
	ENDM	

### Calling Sequence

name	63	13*8
	03	SENTRY
	00	00
	00	logical unit number

## LABEL

LABEL	MACRO	(L, N, E, R, C)
	CALL36	(14)
	00	location of name (N)
	00	logical unit number (L)
	VFD	A24/C, A12/E, A12/R
	ENDM	

### Calling Sequence

name	63	14*8
	03	SENTRY
	00	location of name
	00	logical unit number
	VFD	retention code or date written
	VFD	edition number, reel number

## SAVE

SAVE	MACRO	(L, P)
	CALL36	(15)
		unsave parameter (P)
	00	logical unit number
	ENDM	

### Calling Sequence

name	63	15*8
	03	SENTRY
	00	0
	00	logical unit number

UNSAVE	MACRO	(L)
	SAVE	( (L), 77B)
	ENDM	

### Calling Sequence

name	63	15*8
	03	SENTRY
	00	77B
	00	logical unit number

## INTERRUPT

### SELECT

```
SELECT  MACRO      (F, I)
        CALL36    (16)
        IFT, EQ   F, /I, 2
        10       **
        00       I
        IFT, NE   F, /I/, 2
        00       **
        Z         (F, I)
        ENDM
```

#### Calling Sequence

```
name    63        16*8
        03        SENTRY
        00        **
        F         interrupt address
F = 01   interrupt on shift fault
        02   interrupt on divide fault
        03   interrupt on exponent overflow fault
        04   interrupt on exponent underflow fault
        05   interrupt on arithmetic overflow fault
        10   interrupt on storage reference fault
        11   1604 mode alert
        12   trace mode alert
        14   interrupt on illegal instruction fault
        15   interrupt on operand parity fault
        16   manual interrupt alert
        17   abnormal termination
```

### REMOVE

```
REMOVE  MACRO      (F)
        CALL36    (17)
        VFD       A6/F, O18/0
        ENDM
```

#### Calling Sequence

```
name    63        17*8
        03        SENTRY
        00        **
        F         00
```

## LIMIT

LIMIT	MACRO	(D, R, I)
	CALL36	(18)
	IOSR	time (D)
	00	reject address (R)
	00	interrupt address (I)
	ENDM	

### Calling Sequence

name	63	18*8
	03	SENTRY
	IOSR	(seconds, milliseconds)
	00	reject address
	00	interrupt address

## BOUND

BOUND	MACRO	(LB, UB, R, I)
	CALL36	(21)
	XMIT	LB, UB
	00	reject address
	00	interrupt address
	ENDM	

### Calling Sequence

name	63	21*8
	03	SENTRY
	XMIT	lower bound, upper bound
	00	reject address
	00	interrupt address

## UNBOUND, EXIT, FREE, TIME, DATE, HERESAQ

UNBOUND	MACRO	22
EXIT		
FREE		
TIME		
DATE		
HERESAQ		
	CALL36	00
		19
		20
		23
		27
	ENDM	

Calling Sequence

name	63	22	
		00	
		19	*8
		20	
		23	
		27	
	03		SENTRY

**SPECIAL**

**LOADER**

LOADER	MACRO	
	CALL36	(24)
	ENDM	

Calling Sequence

name	63	24*8
	03	SENTRY

**LIBRARY**

LIBRARY	MACRO	(L, R, NA, NU)
	CALL36	(25)
	00	reject address (R)
	00	logical unit number (L)
	VFD	A24/record number (NU)
	00	record name address (NA)
	ENDM	

Calling Sequence

name	63	25*8
	03	SENTRY
	00	reject address
	00	interrupt address
	VFD	record number
	00	record name address

## MEMORY

MEMORY	MACRO	(B, LL, UL)
	CALL36	(26)
	VFD	O6/0, B3/B, A15/UL
	00	LL
	ENDM	

### Calling Sequence

name	63	26*8
	03	SENTRY
	VFD	bank designator, upper limit
	00	lower limit

## CORE

CORE	MACRO	(LL, UL)
	MEMORY	(* , LL, UL)
	ENDM	

# SCOPE MESSAGES AND DIAGNOSTICS

# C

---

Messages are written to the operator on the operator comment medium (OCM), to the programmer on the standard output unit (OUT) and standard punch unit (PUN), and to the installation on the accounting unit (ACC). Messages do not require any corrective action.

Diagnostics are indications of error conditions which require corrective action. They are written on OCM or OUT, or both.

## MESSAGES ON OUT

message	meaning
AET printouts	The AET table is printed according to the AET statement options.
BEGIN JOB AT hhmm - ss	The time is printed on OUT when no sequence statement is present.†
SCOPE c. xy	To separate the control statements from the run and identify the version of the system. c refers to a specific COSY tape. x is the field update number. y is the installation update number.
control cards except JOB, SEQUENCE, ENDScope	SCOPE control cards are listed before being processed.
END JOB SEQUENCE xxxxxx DATE mm/dd/yy TIME hhmm - ss ELAPSED TIME xx HRS xx MIN xx SEC	Job termination message.††
end-of-file mark	Jobs on OUT are separated by an EOF record.
JOB card image	The JOB card is printed.

---

† hh = hours      †† mm = month  
mm = minutes      dd = day  
ss = seconds      yy = year

message

meaning

SEQUENCE card image with words 8, 9, and 10 preset to: AT hhmm - ss

The SEQUENCE statement and job initiation time are written.

1

A page eject record separating runs.

The following example illustrates the messages and information written to the programmer on OUT for a typical compilation and execution. According to the <sup>7</sup>FTN, L, A, X card, the FORTRAN source program is listed, followed by the assembly language listing; next, a COMPASS program is assembled and listed. All SCOPE control statements, times, MAP, and program output are listed. A memory dump of the program, labeled and numbered common, and console scoop are forced because of an illegal jump out of bounds.

SEQUENCE .8  
 JOB.52462, D D STONE .3  
 SCOPE 6.00  
 FTN.L.A.X  
 (page eject)

SCOPE control statements

AT 2023 - 45

hhmm - ss

**FORTRAN Source Program Listing**

5.1

PAGE NO. 1

```

PROGRAM A TEST RECOVERY OPTIONS
TYPE INTEGER 8, SUM
COMMON/A/B(10)
DATA (B=10,16.4,2,3,5,7,9,2,18)
PRINT 12
12  FORMAT (17H SUM = B1 + B1+1)
DO 7 I=1,9,2
SUM=B(I)+B(I+1)
PRINT 10,SUM,B(I),B(I+1)
10  FORMAT (315)
7   CONTINUE
CALL START
END
0   0   22002 ← compiler diagnostic
    (page eject)
    
```



Assembly Language Listing of FORTRAN Subprogram

2/17F

ED 0

PAGE NO.

2

```

PROGRAM LENGTH          00055 IDENT TESTRECX
ENTRY POINTS
BLOCK NAMES            TESTRECX 00007
EXTERNAL SYMBOLS
    ELD.
    Q8QDICT.
    START
    STH.
    Q8QENTRY
00000 63 0 P00000 EXIT. 63 ($)*
        20 0 X77777          20 ($)Q8QDICT.
00001 00 0 00000 DICT. OCT 000000000000000000
        00 0 00000
00002 63 2 56263          OCT 6325626351252367
        51 2 52367
00000          A BLOCK 10
00000          COMMON B(10)
        C00000          ORGR B
00000 00 0 00000          OCT 0000000000000012
        00 0 00012
00001 00 0 00000          OCT 0000000000000020
        00 0 00020
00002 00 0 00000          OCT 0000000000000004
        00 0 00004
00003 00 0 00000          OCT 0000000000000002
        00 0 00002
00004 00 0 00000          OCT 0000000000000003
        00 0 00003
00005 00 0 00000          OCT 0000000000000005
        00 0 00005
00006 00 0 00000          OCT 0000000000000007
        00 0 00007
00007 00 0 00000          OCT 0000000000000011
        00 0 00011
00010 00 0 00000          OCT 0000000000000002
        00 0 00002
00011 00 0 00000          OCT 0000000000000022
        00 0 00022
        P00003
00003          FORMAT. ORGR *
        BSS 4
        ENTRY TESTRECX
00007 63 0 00000 TESTRECX UBJP ($)*,*,*
        01 0 P00007
00010 63 0 P00007 + 63 ($)*-1
        20 0 X00000          20 ($)Q8QDICT.
.
.
.
00052 75 0 P00050          SLJ BEGIN.
        00 0 P00001          OO DICT.
00053 50 1 00000 ENDING. ENI 0,1
        75 0 P00000          SLJ EXIT.
        END TESTRECX
NULLS TESTRECX .7

```

(page eject)

COMPASS Program

2/17F

ED 0000 PAGE NO. 1

PROGRAM LENGTH	00004	IDENT	RECOV	
ENTRY POINTS				
BLOCK NAMES	START 00000			
	00031			
	7070 00012			
	LABELED 00012			
		ENTRY	START	
			THIS PROGRAM ATTEMPS A JUMP OUTSIDE ITS AREA OF MEMORY, PRODUCING A BOUNDS FAULT TERMINATION.	
00000		BLOCK	0	
00000		COMMON	C(5,5)	
00000	7070	BLOCK		
00000		COMMON	A(10)	
00000	LABELED	BLOCK		
00000		COMMON	B(10)	
00000	50 0 00000	START	NO	
	50 0 00000			
00001	50 1 00001 +	ENI	1,1	
	50 2 00002	ENI	2,2	
00002	50 3 00003	ENI	3,3	
	04 0 77777	ENQ	77777B	
00003	63 0 00000	63	0	} jump out of bounds
	03 0 00076	03	62	
		END		
NULLS		START	C	A
				B

(page eject)

LOAD ← SCOPE control statements  
 RUN,2.1000.7.,, FOR STANDARD OUTPUT UNIT LISTINGS  
 (page eject)

MAP

PROGRAM NAMES											
1 77722	TESTREX	00055	1 77704	RECOV	00004	1 77577	Q8QENTRY	00073	1 77171	IOS.	00406
1 77006	Q8QERROR	00163	1 75632	IOP.	01154	1 75463	ALLOC.	00147	1 75404	STH.	00057
1 73746	IOH.	01436									

PROGRAM EXTENS.  
 NONE

LABELED COMMON				
1 77710	A	00012	1 77672	LABELED 00012

NUMBERED COMMON				
1 00001		00031	1 00032	7070 00012

ENTRY POINTS							
0 00062	SENTRY	1 77731	TESTREX	1 75020	ELD.	1 77577	Q8QDICT.
1 77704	START	1 75410	STH.	1 77600	Q8QENTRY	1 77651	EXIT
1 75635	IOP.	1 77244	Q8QHIST.	1 77171	IOE.	1 77363	Q8QCHAIN
1 77211	IOS.	1 77503	IOR.	1 77006	Q8QERROR	1 75543	ALLOC.
1 75470	RETURN.	1 75602	ALLOC IN.	1 73746	IOH.		

EXECUTION STARTED AT 2024 -17 ← a FORTRAN run time printout  
 (page eject)

Program Output

SUM = B1 + B1+1  
 26 10 16  
 6 4 2  
 8 3 5  
 16 7 9  
 20 2 18

(page eject)

Recovery Dump

RECOVER ILL. BOUNDS INT

(ZERO)=6300000101177707 SENTRY=6300000001000062 BOUNDS=0017777700014600

A = 0000000000000001 Q = 7777777777777777 D = 0000000000000000 IR = 0000000000000000 IM = 7777777777771600
B1 = 00001 B2 = 00002 B3 = 00003 B4 = 64003 B5 = 00001 B6 = 00000 MS = 0000000000000014

100001 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
100006 5 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00020640 00000006
100013 12 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
\*\* \*
100025 24 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
7070
100032 0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
100037 5 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
IOH.
173746 0 UBJP 6300000101175430 RXT00741225 STA20075261 SAL61074210 SAL61074356 SAL61074430 RXT00740407
173752 4 SAU60074207 SIU56174255 SIL57274255 SIU56374256 SIL57474256 SIU56574257 SIL57674257 ENI50100010
173756 10 STQ21075257 ENI50000000 SBYT 6301170350574206 SBYT 6301170350574213 ENI50200021 ENI50500000
173762 14 SBYT 6302000150774210 RXT00740653 RXT00740343 SBYT 6301170350574177 SBYT 6300301750475262
173766 20 STA20075344 LDA12073746 STA20074261 INA11000001 STA20075031 INA11000001 SBYT 6300002250575162
. . . . .
175376 1430 LDQ16715312 ROP00000000 SST40000000 ROP00000000 SAU60000000 ROP00000000 SAU60606060 SAU60606000
175402 1434 SAU60606060 SAU60604000 RSW00700000 ROP00000000
STH.
175404 0 XMIT 6317540420177577 UBJP 6300000101177753 FSB31614660 62633033 XMIT 6317540420177577
175410 4 UBJP 6300000101177763 XMIT 6317541020175405 SBYT 6300001750775410
175414 10 ENQ04000000 ENI50000000 BRTJ 6300000003177211 SAL61075434 LDQ16075422 LBYT 6300302250375405
. . . . .
175454 50 LDA12076156 AJP22075453 ALS05000017 ENI50000000 RAD70000442 SLJ75075447 ROP00076212 LIU52575457
175460 54 SIU56575466 ENA10000001 STA20076063 STA20076153 SLJ75075467 ENI50000000
ALLOC.
175463 0 ENA10077777 ENI50000000 ENI50100075 ENI50200017 ENI50300000 ENI50464003 ENI50500002 ENI50600052
175467 4 LDQ16075602 RXT00740615 UBJP 6300000101176770 RTJ 7701440075475606 RGJP 6200000001075464
. . . . .
175623 140 BRTJ 6307777003000062 ROP00214343 SBL46232163 SBL46516023 FAD30213145 SAU60234346 AJP22222551
175627 144 DV125246060 SAU60606060 FAD30000004 ROP00175624 77700000 INF77000000
IOP.
175632 0 ENI50100011 ENI50200047 ENI50300000 ENI50464003 ENI50500002 ENI50600052 UBJP 6300000101175436
175636 4 STA 7710440020076173 SIU56175632 SIL57275632 SIU56375633 SIL57475633 SIU56575634 SIL57675634
. . . . .
177663 64 SBL46456062 63215163 DV125246021 63606060 QRS02000204 SAU60400107 SAU60606060 SAU60606060
177667 70 ROP00000005 ROP00177662 ROP00000001 ROP00177671 ARS01606060 SAU60606060
LABELED
177672 0 00000000 00000020 07400077 00000074 00000007 00000000 00000000 00007000 40000000 00000011
177677 5 10400000 00000074 77777777 00000000 60606060 00000000 00000100 00000074 77777777 77700350
RECOV
177704 0 UBJP 6300000101177767 ENI50100001 ENI50200002 ENI50300003 ENQ04077777 BRTJ 6300000003000076
A
177710 0 00000000 00000012 00000000 00000020 00000000 00000004 00000000 00000002 00000000 00000003
177715 5 00000000 00000005 00000000 00000007 00000000 00000011 00000000 00000002 00000000 00000022
TESTRECX
177722 0 XMIT0631777222 177577 UBJP 6300000101177652 LBYT 6325626351252367 CONN 7401073060606264
177726 4 LDL44601360 AJP22316020 SAU60223120 ARS01346060 CONN 7403310534606060 BRTJ 6300000003000062
177732 10 XMIT 6317773120177577 XMIT 6317773120177723 SIU56177775 RTJ75477772 ENA10000075 ENQ04077743
177736 14 BRTJ 6300000103175410 SLJ75077741 ARS01077723 ROP00177725 ROP00000000 BRTJ 6300000103175020
177742 20 SLJ75077743 ROP00077723 ENA10000001 RXT00741225 STA20077776 ENI50000000 RXT00741225 LIL53177776
177746 24 LDA 7710440012177707 ADD14177710 RXT00741225 STA20077771 ENA10000075 ENQ04077763 ENI50000000
177752 30 BRTJ 6300000103175410 SLJ75077755 ARS01077723 ROP00177730 ROP00000000 LDA12077771 77700000
177756 34 LDA 7710440012177707 77700000 LDA12177710 77700000 ENI50000000 BRTJ 6300000103175020
177762 40 SLJ75077763 ROP00077723 ENA10000002 RXT00741225 RAD70077776 INA11077765 AJP22377745 ENI50000000
177766 44 BRTJ 6300000103177704 SLJ75077770 ROP00077723 SLJ75077775 ENI50000000 ROP00000000 ROP00000024
177772 50 SLJ75077735 ENI50000000 BRTJ 6300000103177600 SLJ75077772 ROP00077723 ENI50100017 SLJ75077722
177776 54 ROP00000000 ROP000000013
END JOB SEQUENCE 8 DATE 02/20/64 TIME 2024 - 53 ELAPSED TIME 00 HRS 01 MIN 07 SEC

(page eject)

hhmm - ss

## MAP

MAP is obtained after the program is loaded. It is printed on OUT. Following is the map from the sample job. All information in the MAP is in octal.

Under program names, names declared in the subprograms are listed first, library subroutine names follow. Similarly, entry point names such as TESTRECX and RECOV in the subprograms precede library subroutine entry point names. The library subroutines Q8QENTRY, IOS., Q8QERROR, etcetera, are called by the FORTRAN object program.

### MAP

bank containing program or common block

first word address in specified bank

name of program

size of program

```

PROGRAM NAMES
1 77722 TESTRECX 00055 1 77704 RECOV 00004 1 77577 Q8QENTRY 00073 1 77171 IOS. 00406
1 77006 Q8QERROR 00163 1 75632 IOP. 01154 1 75463 ALLOC. 00147 1 75404 STH. 00057
1 73746 IOH. 01436

```

PROGRAM EXTENS.  
NONE

block name

size of block

```

LABELED COMMON
1 77710 A 00012 1 77672 LABELED 00012

```

blank numbered common

size

```

NUMBERED COMMON
1 00001 00031 1 00032 7070 00012

```

entry point name

```

ENTRY POINTS
0 00062 SENTRY 1 77731 TESTRECX 1 75020 ELD. 1 77577 Q8QDICT.
1 77704 START 1 75410 STH. 1 77600 Q8QENTRY 1 77651 EXIT
1 75635 IOP. 1 77244 Q8QHIST. 1 77171 IOE. 1 77363 Q8QCHAIN
1 77211 IOS. 1 77503 IOR. 1 77006 Q8QERROR 1 75543 ALLOC.
1 75470 RETURN. 1 75602 ALLOCIN. 1 73746 IOH.

```

entry point address

## RECOVERY DUMPS

Recovery dumps are taken for programs which do not terminate normally. The dump is written on OUT. There are 4 computer words per line in octal with mnemonics for program locations, and 5 computer words per line in octal for common. An octal console scoop is always taken. A recovery key of seven will dump all of storage except SCOPE. The dump will be blocked on tape.

A recovery diagnostic (see diagnostics on OUT) is written indicating the cause of the abnormal job termination.

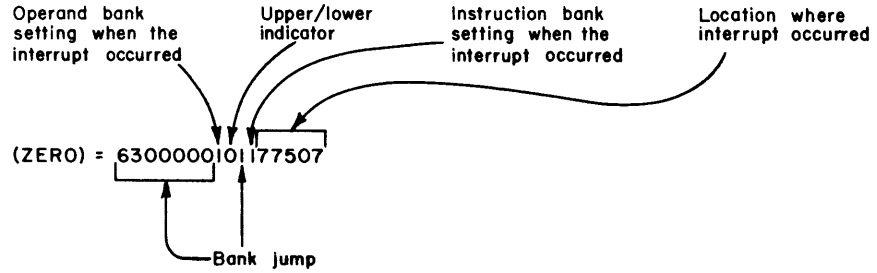
Additional identical lines are omitted and are indicated by a row of asterisks.

Mnemonics in the console scoop are:

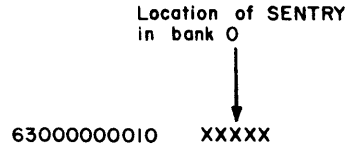
A	A register
Q	Q register
D	D register
IR	Interrupt register
IM	Interrupt mask register
B1-6	Index registers 1-6
MS	Miscellaneous mode selections register
BOUNDS	Bounds register
(ZERO)	Contents of all (0)00000 at recovery time
SENTRY	Contents of SENTRY at recovery time

If the dump occurs with the interrupt system active, location zero contains an unconditional bank jump (63.0) to the location +1, where the interrupt occurred.

**Example:**



SENTRY will contain the following:



If the dump occurs during interrupt lockout mode, it is because of an I/O request causing abnormal termination and SENTRY contains an unconditional bank jump to the location +1 of the I/O request.

When autoload recovery is used, (ZERO) and SENTRY contain zero.

The recovery dump for the typical job is shown.



```

10S. ← name of library subroutine
177171 0 ROP00000000 ROP00000001 SAU 7710440060077171 ENA10000000 LLS07000052 DVF 7710000127077573
177175 4 DVF 7710000127077574 SBYT 6300001450777207 LDA12077171 ENQ04000075 BRTJ 6300001503175635
177201 10 BRTJ 6300000003177006 ROP00000000 ROP00000000 ROP00000000 ROP00177204 DV125515146 IN151604623
.
.
.
177571 400 SLJ75077552 EN150000000 ROP00000000 ROP00000000 ROP00000000 ROP00000012 ROP00000000 ROP00000100
177575 404 ROP00000000 ARS01000000 ROP00000000 RXT00777777
QBQENTRY ← name of library subroutine
177577 0 XMIT 6317540420177577 UBJP 6300000101177774 RXT00741225 SLJ75077600 ENA10077600 SAL61077601
177603 4 LDQ16077644 LDA12077600 RXT00740341 ARS01000017 RXT00740565 LDA12100000 RXT00740341 STQ21100000
.
.
.
177667 70 ROP00000005 ROP00177662 ROP00000001 ROP00177671 ARS01606060 SAU60606060
LABLED ← labeled common
177672 0 00000000 00000020 07400077 00000074 00000007 00000000 00000000 00007000 40000000 00000111 ← octal
177677 5 10400000 00000074 77777777 00000000 60606060 00000000 00000100 00000074 77777777 77700350! ← octal
RECOV ← COMPASS program
177704 0 UBJP 6300000101177767 EN150100001 EN150200002 EN150300003 ENQ04077777 (BRTJ 6300000003000076) ← instruction causes interrupt
A ← labeled common
177710 0 00000000 00000012 00000000 00000020 00000000 00000004 00000000 00000002 00000000 00000003 ← octal
177715 5 00000000 00000005 00000000 00000007 00000000 00000011 00000000 00000002 00000000 00000022! ← octal
TESTRECX ← FORTRAN program
177722 0 XMIT 6317772220177577 UBJP 6300000101177652 LBYT 6325626351252367 CONN 7401073060606264
177726 4 LDL44601360 AJP22316020 SAU60223120 ARS01346060 CONN 7403310534606060 BRTJ 6300000003000062
177732 10 XMIT 6317773120177577 XMIT 6317773120177723 SIU56177775 RTJ75477772 ENA10000075 ENQ04077743
177736 14 BRTJ 6300000103175410 SLJ75077741 ARS01077723 ROP00177725 ROP00000000 BRTJ 6300000103175020
177742 20 SLJ75077743 ROP00077723 ENA10000001 RXT00741225 STA20077776 EN150000000 RXT00741225 LIL53177776
177746 24 LDA 7710440012177707 ADD14177710 RXT00741225 STA20077771 ENA10000075 ENQ04077763 EN150000000
177752 30 BRTJ 6300000103175410 SLJ75077755 ARS01077723 ROP00177730 ROP00000000 LDA12077771 77700000
177756 34 LDA 7710440012177707 77700000 LDA12177710 77700000 EN150000000 BRTJ 6300000103175020
177762 40 SLJ75077763 ROP00077723 ENA10000002 RXT00741225 RAD70077776 INA11077765 AJP22377745 EN150000000
177766 44 BRTJ 6300000103177704 SLJ75077770 ROP00077723 SLJ75077775 EN150000000 ROP00000000 ROP00000024
177772 50 SLJ75077735 EN150000000 BRTJ 6300000103177600 SLJ75077772 ROP00077723 EN150100017 SLJ75077722
177776 54 ROP00000000 ROP00000013
END JOB SEQUENCE 8 DATE 02/20/64 TIME 2031 - 51 ELAPSED TIME 00 HRS 01 MIN 07 SEC

```

absolute octal 18 bit address      octal address relative to the beginning of the subprogram or common block. When a new name is printed, the relative address is reset to 0.      word contents

octal with mnemonics



## SNAP AND TRACE

SNAP and TRACE dumps are in the same format. The dump consists of a console scoop and a memory dump written on OUT.

In the console scoop:

A and Q registers are printed in the requested mode.

Index registers, the Bounds register and the D register are printed in octal.

Interrupt register, interrupt mask register and sense switches are printed in binary. Above the interrupt registers are bit positions for reference purposes.

In the memory dump, 4-10 computer words appear per line depending on the mode requested; asterisks in locations indicate that the information is identical to the preceding line.

The example shows the program and the SNAP dumps. Note the location of the SNAP card images; the first is printed after the LOAD card and the rest after the MAP.

When tracing is initiated, the following statement is written on OUT:

```
TRACING BEGIN AT 6 octal digit address - first word address of area to trace jumps -a1
```

Before each dump in the SNAP format, the message is written:

P = address of jump instruction (P) = contents of P

SEQUENCE .9  
JOB .91001.LARSON.2  
SCOPE 6.00  
COMPASS.X.L  
(page eject)

AT 2031 - 51

2/17F

02/20/64

ED 0

PAGE NO. 1

PROGRAM LENGTH				IDENT	SNAPSSS
ENTRY POINTS				30011	
BLOCK NAMES					
			FIRST	30001	
			HELP	30006	
		1234		00144	
00000				ENTRY	FIRST,HELP
00000			1234	BLOCK	0
00000				COMMON	SPACE(100)
30001	50	0	00000	ARY	30001B
	50	0	00000	FIRST	NOP
30002	50	1	00000		NOP
	10	0	00000	PG	0.1
30003	20	1	P00000		0
	50	0	00000	STR	ARY.1
30004	54	1	30000		
	75	0	P30003	ISK	30000B,1
30005	10	0	77777	SLJ	STR
	50	0	00000	ENA	77777B
30006	20	1	P00000	HELP	ARY.1
	50	0	00000		
30007	54	1	30000	ISK	30000B,1
	75	0	P30006	SLJ	HELP
30010	75	0	P30001	SLJ	FIRST
	50	0	00000		
				END	FIRST
				SPACE	PG

NULLS  
(page eject)

LOAD,69  
SNAP,HELP,SNAPSSS,+550.OC.1.7.3.RESULT  
(page eject)

MAP  
(page eject)



## OCTAL CORRECTIONS AND LOADER CARDS

The following example contained an illegal instruction. OCC cards were used to correct the program and inserted into the binary deck. Note, the program extension area is included in the MAP.

All loader control cards (LCC) are written on OUT. These include BANK, MAIN, OVERLAY, and SEGMENT cards.

```
SEQUENCE,931
JOB,52462,DDS,5
SCOPE 6.00
BANK,(1),HELP,/12347
OCC IN PROGRAM USELESS 00003+ 75000000E
OCC IN PROGRAM USELESS 00000E 77200000 20000004+75000000+
(page eject)
```

AT 1409 - 45

entry point in USELESS  
common block in USELESS

} Octal corrector cards are  
inserted into a binary deck.

```
PROGRAM NAMES
1 77771 USELESS 00006
```

```
PROGRAM EXTENS.
1 77767 USELESS 00002
```

} Second octal correction is loaded  
in program extension area.

```
LABELED COMMON
NONE
```

```
NUMBERED COMMON
1 00001 1234 00500
```

```
ENTRY POINTS
0 00062 SENTRY
RUN,4,1000,1 1 77771 HELP
```

Messages are sent on the input comment medium (ICM) to the computer when SCOPE requests information. The comments may be chosen by the operator. He may abandon the job, begin at any sequence, or press manual interrupt.

## MESSAGES ON ICM

Message	Meaning
AET	An AET statement as described in this manual may be typed. (See p. 2-22.)
ENDSCOPE	Log off the last job, and perform final accounting. If the computer is in Satellite mode, the units are turned over to SASY to process, otherwise, the system units are unloaded and the computer stops.
EQUIP	An EQUIP statement as described in this manual may be typed. (See p. 2-3.)
LOOK	Reply to a computer request for a tape. The operator should mount the tape, and enter the message on ICM. The computer performs a label search to find the tape.
NEXT or NEXT, c or NEXT, c+d	c = job sequence number of a SEQUENCE card d = the positive decimal number of jobs to be skipped  If neither c or d are given, the computer executes the next consecutive job. If c only is given, the computer searches for a job containing a SEQUENCE card with the job sequence number, c. If c + d is given, the computer finds the job sequence number, c, on a SEQUENCE card, skips the number of jobs indicated by d and executes the succeeding job.
NONE	Reply to a computer request for a tape. The operator enters the message and the computer attempts to process the job without the requested tape.
OK and carriage return	Continue processing.
PAUSE or PAUSE, job sequence number	Transfer control to the operator at the beginning of a computer job when the SEQUENCE statement is encountered. If no job sequence number is given, the computer halts at the next encountered SEQUENCE statement; otherwise the computer halts at the SEQUENCE statement containing the given job sequence number. The type-in light turns on when this halt occurs.

Message	Meaning
REPEAT	Log off the current job, find its beginning and start it over.
... and carriage return	Transfer control to the programmer's manual interrupt subroutine for the currently running program.

No available input reel, duplicate input reels, no available reel for output, or non-standard label read assignment (unlabeled tapes) may prevent the normal assignment of logical units. In all of these cases, the operator is queried on the OCM unit and must reply on the ICM unit.

If there is no available input reel - a reel with the specified label cannot be found by SCOPE, the message on OCM is nneerr, NEEDED. He may repeat the job (REPEAT message on ICM), or skip to the next job (NEXT). He may mount the reel and instruct SCOPE to search for the desired label again (LOOK). He may attempt to force the assignment (EQUIP). This attempt is futile if the label does not check, consequently, the contents of the label will be typed out. If the tape assignment is requested by a RDLABEL request, he may type a message (NONE) stating that the tape is not available. This will set the first word of the user's buffer to +0 and return control to the programmer.

If there are duplicate or multiple input reels (reels with identical labels) and the request forcing the assignment is not RDLABEL, the message on OCM is: nneerr, name of tape, NEEDED. If the request is RDLABEL, SCOPE assigns one of the tapes to the specified logical unit, types a message telling the operator which unit is assigned and transfers control to the programmer.

When no reel is available for output, "CANNOT ASSIGN uu = hh oo" is typed. The operator may attempt to force the assignment with the statement EQUIP u = MTnn. The label for physical unit MTnn is typed out and the operator is queried. He may now reply on ICM with NEXT, REPEAT, EQUIP, u = MTnn (attempt to force the assignment to a different physical unit), or OK.

SCOPE assigns output tapes with non-standard labels in the same manner as standard labeled tapes. The operator is queried for the assignment of input reels unless they were previously assigned by an EQUIP statement.

The following typewriter character codes are for comments typed in by the operator:

TYPEWRITER  
CHARACTER CODES

CHARACTER	BCD		TYPE†	CHARACTER	BCD		TYPE†
0	00		43	-	40		04
1	01		77	J	41		70
2	02		37	K	42		15
3	03		33	L	43		41
4	04		47	M	44		72
5	05		57	N	45		31
6	06		13	O	46		42
7	07		57	P	47		54
8	10		17	Q	50		10
9	11		07	R	51		56
=	13		34	S	53	U	47
@	14	U	37	*	54	U	17
:	15	U	50	blank	60	I	60
;	16		50	/	61		44
?	17	U	44	S	62		46
+	20	U	34	T	63		75
A	21		12	U	64		35
B	22		01	V	65		36
C	23		11	W	66		02
D	24		55	X	67		71
E	25		51	Y	70		40
F	26		30	Z	71	I	73
G	27		74	,	73		14
H	30		45	(	74	U	07
I	31		16	tab	75	I	62
.	33	I	32	backspace	76	I	61
)	34	U	43	carriage return	none	I	63
				upper case	none	I	64
				lower case	none	I	66
				quotation mark	none	I	52

†Alphabetic characters are not typed in upper case, but the code is transmitted. I signifies that the upper case and lower case characters are identical. U signifies that upper case must be selected for this particular character.

## MESSAGES ON OCM

Message	Meaning	Action
AET printouts	AET table is printed according to AET statement options.	
ENTER DATE MDY	Initial autoloading or computed time is later than midnight.	Enter date.
JOB ABANDONED	Job is terminated because of control statement errors on INP.	
ssssss , iiiiii,	New job. The first 6 characters of sequence card and JOB identification are typed.	
PAUSE xxxxxx	A pause was requested before this job.	Any legal operator statement. A period means continue with next job.
RELEASED hh oo	hh oo is released.	
UNLOADED hh oo PRINT hh oo PUNCH hh oo PR/PU hh oo	Physical unit hh oo is unloaded. Tape is to be saved and has been unloaded.	
nneer, NEEDED	No available input reel.	Assign a tape.
nn eerr, = hh oo†	Identifies a numbered unit.	
nneerr, name of tape, NEEDED	Duplicate or Multiple input reel.	SCOPE forced assignment of a reel.
nn eerr, name of tape 14 characters = hh oo	Identifies a named unit.	
nn eerr, (TAPE IS UNLABELED)	Identifies an unlabeled unit.	
MToo = contents of label	Response to EQUIP, nn=MToo informing operator of label on MToo.	If tape is acceptable to operator, type OK. or a period. If tape is not acceptable, reassign or abandon the job.

† nn = logical unit number  
ee = edition number appearing in label, or blank

rr = reel number  
hh = hardware type mnemonic  
oo = AET ordinal of unit, within hardware type





## AET LISTING

The AET listing on OCM or OUT can be obtained by entering the statement AET, , or AET, , OUT, on either OCM or INP.

### Example:

Ordinal:	Entry:
AET 001	0160100100000400
AET 002	0160100100000600
AET 003	0160100101401000
AET 004	0160100101401200
AET 005	0160100102001400
AET 006	0160100102001600
AET 007	0160100102002000
AET 010	0160100102002200
AET 011	0160100101002400
AET 012	0160100100002600
AET 013	0160100100403000
AET 014	0160100100003200
AET 015	0160100100003400
AET 016	0160100100003600
AET 017	0160100101404000
AET 020	0160100100060003
AET 021	0440500200040004
AET 022	0521200301520002
AET 023	0621400400340000
AET 024	0631401000520001
AET 025	1060000503020205
AET 026	2063040607000405
AET 027	2122000707000205
AET 030	2142000707000205

## MESSAGES ON ACC

Message	Meaning
END JOB SEQUENCE xxxxxx DATE mm/dd/yy TIME † hhmm - ss ELAPSED TIME xx HRS xx MIN xx SEC	Job termination message.
BEGIN JOB AT hhmm - ss	Time is printed on ACC when no sequence statement is present.†
JOB card image	The JOB card is printed.
SEQUENCE card image with words 8,9, and 10 preset to: AT hhmm - ss	The SEQUENCE statement card image and job initiation time are written.

---

† hh = hours	mm = month
mm = minutes	dd = day
ss = seconds	yy = year

## MESSAGES ON PUN

Message	Meaning
end-of-file mark	Separator for punch tape at the end of the job in which unit 62 was used.
SEQUENCE, ssssss oo. . . o (80 character record)	Separator for punch tape at beginning of each job which requires unit 62. This record is binary.

## DIAGNOSTICS ON OUT

Diagnostic	Condition	Action
CONTROL STATEMENT FORMAT ERROR	<ul style="list-style-type: none"> <li>a) Illegal BCD character on a control card.</li> <li>b) Illegal equivalence declaration.</li> <li>c) Illegal logical unit number in the FILE LOAD or EQUIP card.</li> <li>d) Illegal parameter on RUN card.</li> </ul>	Correct the illegal field.
CONTROL STATEMENT REJECTED	Statement out of sequence. Illegal character on an ENTRY POINT NAME card.	Check control cards for proper ordering.
ERRORS IN LOADING I/O DRIVERS	Loading errors detected when loading drivers.	Consult loader diagnostics and correct I/O driver routines on library tape.
ERRORS RETURNED - LOADING NOT ATTEMPTED	Compiler errors in the job before this loading operation was attempted. The LOAD statement precedes this message.	Correct compiler/assembler errors.
ERRORS IN LOADING OPERATION	Loader returned errors.	Consult loader diagnostics.
FILE I/O OR DATA ERROR	I/O error in FILE statement or illegal control statement within data to be filed.	Check data to be filed.
TIME LIMIT EXCEEDED	Time Limit exceeded while reading a control statement.	Extend specified time limit.

Diagnostic	Condition	Action
TAPE READ ERROR ON INP	1) Parity errors on control card read. (May be a BCD record.) 2) Parity errors on binary record when skipping.	Check if program read all of data.
TRA = 0	Transfer address after loading is zero. No transfer name provided on TRA card (generated by COMPASS END cards). Job is abandoned.	Provide a transfer name.

## RECOVERY DUMP DIAGNOSTICS

Diagnostic	Condition
BAD SENTRY CALL	Entry to SENTRY through a jump during interrupt mode, not necessarily through a call.
CWA ILLEGAL	Control word address is zero.
EOF ON SYST UNIT	Attempt to backspace past end-of-file or read past end-of-file on INP.
ILL. BOUNDS INT	Illegal bounds interrupt
ILLEGAL INSTR.	Illegal instruction
ILLEG LABEL CALL	Illegal LABEL request, (labeling a scratch tape; or edition number or reel number too large; or more labels than tapes).
MEMORY REQUEST	Illegal MEMORY request, (setting limits in an illegal bank).
ILLEG MODE CALL	Illegal MODE request, (specifying contradictory modes within a call).
ILL. USE OF UNIT	Illegal use of unit, (a label macro reference to a unit which is not a tape).
IMPOS FUNCT	Unit not capable of function, (write on card reader or any unit specified as RO).
INTERNAL REJECT	Hardware problem, see 3600 Reference Manual, Appendix II.
INT. ADDR ILLEG	Interrupt address is 0 or -0.
INT. ADDR. = 0	Interrupt address of a BOUND, SELECT, or LIMIT request is zero.

Diagnostic	Condition
INT. FEATURE BAD	Interrupt in SELECT request is illegal (0 or > 17).
LIBRARY REQUEST	Illegal condition in library request (table of contents not available for requested unit).
L. U. N. ILLEGAL	Logical unit number illegal, negative, zero, or greater than 80.
MACHINE OR SYSTEM ERROR FROM (5 digit address within SCOPE)	SCOPE equipment tables destroyed; machine or equipment malfunction.
MEMORY REFERENCE	A reference has been made to a non-existent bank.
NO DRIVER	Equipment requiring driver was not properly assigned.
NO WRITE ENABLE	Trying to write on a tape with no ring which was previously read.
OPERAND PARITY	Hardware problem, see 3600 Reference Manual, Appendix I.
OPERATOR TERM.	Operator terminated job with an operator statement on OCM or recovery autoloading.
PRINT LIMIT	Print limit specified on RUN card exceeded.
REJECT ADDR = 0	Reject address must be specified.
REQUESTED DUMP	TD was specified on the RUN card and the job terminated normally.
TIMESUP	Time limit specified on JOB or RUN card exceeded.

### SNAP/TRACE DIAGNOSTICS

If an error occurs while processing SNAP or TRACE cards, the following message is written on OUT followed by the card image.

```

CARD   yyy { SNAP } ERROR AT COLUMN xxx mmmmmmmm
          { TRACE }
          yyy      number of card
          xxx      column being processed when error was detected
          mmmmmmmm diagnostic listed below

```

Diagnostic		Condition
BIG	NUM	Octal value more than 6 digits
BIG	PAR	Parameter greater than 4096
EXC	CARD	Insufficient information on SNAP or TRACE card
FWA		FWA exceeds LWA
ILL	MODE	Illegal character in mode field
ILL	OCT	Illegal character in octal field
ILL	PAR	Illegal character in parameter field (non-numeric)
ILL	SLSH	Slash appears illegally
LOC	MEM	Location references non-existent memory
LOC1	OCT	Initial location is absolute octal
LOC1	REL	Relative location with no preceding name
LOC2	OCT	LOC2 absolute octal
NO	LOC2	No LOC2 on TRACE card
NO	LWA	No LWA
NO	ROOM	Not enough available memory in bank for DEBUG tables
NOT	LOAD	Name not in loader tables
REL	ERR	Relative address with no preceding name
LOC1	BIG	LOC1 greater than/equal to LOC2 on TRACE card

## LOADER DIAGNOSTICS

All loader diagnostics are written on OUT in the following format:

LOADER ERROR P<sub>1</sub> P<sub>2</sub> P<sub>3</sub>

When the loader detects an error in the loading operation, the diagnostic is written on OUT, and loading continues. For example, after writing the checksum error diagnostic on OUT, the card containing the error is processed as though the checksum was correct. The exceptions are the memory overflow errors and too many overlay tapes which cause the loading operation to be terminated. When errors are encountered during the loading operation, no library subroutines are loaded.

P<sub>1</sub> is the last subprogram name encountered by the loader; or if the errors are on the library tape, P<sub>1</sub> will be LIB yyy, or blank. The values of P<sub>2</sub> and P<sub>3</sub> for each error are shown below.

P <sub>2</sub>	P <sub>3</sub>	Condition
BAD LAT	Address from LAT string.	More than 32767 entries in LAT string.
BANK	Portion of BANK card containing format error.	BANK card format error.
CARD SEQ	Col. 1 and 2 of current card.	Card out of sequence, e.g., LAT card followed by EXT card.
CHKSUM	Col. 1 and 2 of current card.	Checksum error; Checksum on binary card does not agree with computed checksum.
COM LNG	Name of common block.	Common block length error. Labeled common blocks differ in length or multiple numbered common blocks in one bank vary in length.
CD TERM	Col. 1 and 2 of current card.	Previous card not terminated. On EPT or EXT, the second card for a continued name was not found.
EOT	Suppressed.	An end-of-tape has been encountered in writing an overlay tape.
FEW BRT	Suppressed.	Either the T portion of a BRT has made a reference past the end of the BRT table or the EXT entries exceed the BRT entries.
FEW LAT	Suppressed.	Either the T portion of an LAT has made a reference past the end of the LAT table or the EXT entries exceed the LAT entries.
ILL BYTE	Col. 1 and 2 of current card.	Illegal byte value in R field of RBD card; byte 10 . . 0, which is not used, was encountered.
ILL CHAR	Col. 1 thru 8 of OCC card.	A character which is not octal or blank appears in a correction, or an illegal relocation designator is used.
ILL PNCH	Col. 1 thru 8 of the OCC card.	An illegal punch configuration appears on an OCC.
LAT OV	LAT(T) .	The computed value of T, when entering an LAT into the permanent loader tables is greater than 4095.

P <sub>2</sub>	P <sub>3</sub>	Condition
LAT RNG	Address of attempted reference.	LAT range error; attempt to reference SCOPE in an LAT string.
LIB TP.	Suppressed.	Library tape error on reading (lost data not parity).
LIB EOF	Suppressed.	Library tape end-of-file.
LIB SEQ	Suppressed.	Library tape card sequence error.
LIB PAR	Suppressed.	Library tape parity error.
LOAD ADD	Col. 1 and 2 of RBD card or col. 1 - 8 of OCC.	Illegal load address on RBD or OCC card. Byte for load address specified either a fixed or decremented address on RBD card. OCC load error may be one of following: Blank load address relocation designator specifies: decremented address fixed address numbered common
MD BANK	EPT name.	Multiply defined bank for EPT. Bank specified for EPT on BANK statement does not agree with bank assigned to entry point.
MD C BK	Name of common block.	Multiply defined common bank. Common block has been assigned to two or more banks by BANK statements or automatic assignment of the block followed by a BANK statement.
MD EPT	Entry point name.	Multiply defined entry point. Same entry point name defined at two addresses.
MD P.N.	Suppressed.	Multiply defined program name. Subprogram name encountered more than once.
MD TRA	Illegal transfer name.	Multiply defined transfer name. More than two TRA cards contain same name.
OVERLAY	Overlay number.	Current overlay card violates overlay rule.
OV MEM	Col. 1 and 2 of current card.	Memory overflow. Not enough memory available to assign block of common, subprogram, or program extension.



	P <sub>2</sub>	P <sub>3</sub>	Condition
OVT MEM		Col. 1 and 2 of current card or col. 1 - 8 of OCC.	Memory overflow. Not enough memory available for loader tables.
PARAM		A register parameter for current loader call.	Parameter error. An illegal call to the loader. S=11 call after loader completed previous load request. S=10 or 00 call before loader completed previous load request. S=10 or 00 and QU = 0.
PARITY		Suppressed.	Non-recoverable parity error on loading unit or overlay tape.
SEC LIM		Col. 1 thru 8 of OCC card.	Current value of load address outside program section being corrected. (program section is the portion of the program referenced by a relocation designator of + or 1-9,0 on OCC) .
SEGMENT		Overlay number followed by segment number.	Current segment card has violated a segment rule.
SEQ NO.		Col. 1 and 2 of current card.	Sequence number on an EPT, BCT, EXT, LAT or BRT is out of sequence.
TAPE NO		Requested tape number.	Logical unit on LCC is out of range.
UN COM		Col. 1 and 2 of RBD card or col. 1-8 of OCC.	Undefined common reference. No common block declared for byte value on RBD card or relocation field on OCC card; or associated common block resulted in a COM LNG error.
UN EXT		External symbol.	Undefined external symbol. Reference to an external symbol not defined as entry point to any subprogram loaded or to any library subroutine.
UN TRA		Transfer name.	Undefined transfer name. Symbolic transfer name on one of the TRA cards was never defined as entry point to a subprogram.
127 BLK		Col. 1 and 2 of current card.	More than 126 common block names have been encountered in a subprogram.
5 TPS		Suppressed.	More than 4 overlay tapes requested when preparing overlay tapes.

## PRELIB ERROR DIAGNOSTICS

When PRELIB detects an error, the diagnostic is written on OUT and processing continues as though the error had not been encountered. Exceptions are memory and table overflow errors, ILLEGAL BCD, and ILLEGAL CONTROL CARD; these cause the job to be abandoned.

Message	Condition	Action
BLOCK LENGTHS UNEQUAL	Labeled common blocks of the same name in programs of an ABS record are not the same length.	Make labeled common blocks the same length.
CARD SEQUENCE ERROR	Card out of sequence.	Correct binary input.
CHECKSUM ERROR	Checksum on binary card does not agree with the computed checksum.	Correct punched checksum.
FEWER LATS THAN EXTS	Fewer LATS than EXTS within a subroutine.	
ILLEGAL BCD	Non-Hollerith character after column one on control card. Job abandoned.	Check for extra punches.
ILLEGAL BYTE VALUE	Illegal byte value in R field of RBD card; byte 10. . . 0, which is not used, was encountered.	Check RBD card and IDC which specifies byte length.
ILLEGAL CONTROL CARD	Control statement illegal, or name more than 31 characters, or name as first parameter (library name) on EXTRACT statement instead of number in tape designation field.	Check control cards.
ILLEGAL LOAD ADDRESS	Byte for load address specified a fixed or decremented address on RBD card.	Correct binary input.
MD BANK	Multiply defined bank on EPT or EXT card.	
MEMORY OVERFLOW FROM PROGRAM	ABS or REL subprogram too long. Job abandoned.	Subdivide program
MORE THAN 126 COMMON	More than 126 common blocks defined.	Reduce number of common blocks.

Message	Condition	Action
MULTIPLE TRANSFER NAMES	More than one named TRA card.	Replace extra named TRA cards with unnamed TRA.
MULTIPLY DEFINED EPT	Entry point name already defined.	Probably an extra subroutine in ABS record.
OVERFLOW FROM xxxxxxxx TABLES	Directory or loader tables too long. Job abandoned.	Reassemble PRELIB with expanded tables.
PARITY ERROR	Parity error in reading or writing a unit used by PRELIB.	Remake binary input, or change tape reels on 71 or 50.
PREVIOUS CARD UNFINISHED	Previous EPT or EXT card incomplete.	Correct binary input.
SEQUENCE NUMBER WRONG	EPT, EXT, BCT, or LAT card out of sequence.	Correct binary input.
SUBROUTINE NOT ON TAPE	Name or record number in DELETE, INSERT, REPLACE, or EXTRACT statement not on tape, or tape spaced beyond record.	Check source library tapes.
UNDEFINED COMMON BLOCK	Relocation byte on RBD card refers to undefined common block.	Correct binary input
UNDEFINED EXTS	Undefined external symbols in an ABS record.	Probably a subroutine is missing.
MACRO FORMAT ERROR, name	Probably misplaced card.	Correct deck.
MISPLACED MACRO CONTROL STATEMENT	Possibly missing end statement.	Correct deck.
PARITY ERROR ON MAIN RECORD	Parity error detected by LOADMAIN while reading MAIN record.	Reconstruct overlay tape.
ILL. L. U. N. SPECIFIED (LOADMAIN)	Logical unit number specified on the control card is not ( $0 < n < 50$ ).	Correct control card.
MAIN NOT ON L. U.	MAIN is not the first record of the tape specified by LOADMAIN control card.	Correct control card or reconstruct overlay tape.
MAIN PROG. OUT OF BOUNDS	MAIN record is not compatible with the memory limits.	I/O and system configuration must be identical to that on which the tape was made.

## DIAGNOSTICS ON OCM

Diagnostic	Condition	Action
BOUNDS REJECTED. AUTOLOAD	Perhaps a machine error.	Autoload to try again.
CANCELLED hh oo	Attempted release or unload cannot be accomplished on unit hh oo.	Mount a tape with blank label or unlabeled and give EQUIP statement in the form: EQUIP,uu=MToo.
CANNOT ASSIGN uu=hh oo	No unit containing a blank label is available for output. oo may be blank. oo was not connectable and ready.	Mount tape with blank label. Type LOOK or EQUIP,uu=hhoo where oo is the unit on which the tape was mounted.
CANNOT FIND REQUESTED JOB	A NEXT, C was issued; job containing sequence C cannot be located.	When type-in light comes on, respond with: Period to reprocess abandoned job, or NEXT for another job.
CONTROL STATEMENT FORMAT ERROR	Illegal BCD character on ICM or illegal equivalence declaration.	Correct illegal character declaration.
PRESS AUTOLOAD	Equipment tables have been destroyed.	Autoload.
SCR ORDINAL TOO LARGE FOR RECOVERY	No blank labeled tapes on bank zero. SCR was assigned to a tape on bank 1.	Press GO to continue, or restart with a blank label tape on a unit on bank 0.
SCOPE PARITY	Parity errors on reading RESIDENT or BOOT from system library tape.	Press GO to run with errors, or set A ≠ 0 to try three more reads.
I BEG YOUR PARDON	Illegal statement.	Check spelling, punctuation, etc., and retype statement.
nn BAD	Malfunction on tape nn. Job abandoned.	Call Customer Engineers to check unit nn.
nn eerr, NEEDED nn eerr, name of tape NEEDED	} Either tape with specified label is not mounted or not ready; or multiple tapes with the specified label are available.	Mount or ready tape containing specified label. Type in: a) EQUIP, nm=hhoo; oo is unit containing needed tape. b) LOOK to search again. c) NONE if no tape is available.

Diagnostic	Condition	Action
nn eerr, (UNLABELED) NEEDED	Unlabeled tape specified in a control card on INP.	Type in: EQUIP, nn=hhoo; oo is unit containing needed tape.
TROUBLE ON hh oo	First reference to assigned unit impossible (EQUIP statement referenced non-existent unit). Label did not match.	Operator must reassign the unit.
PUT RING IN hh oo	Tape with no write enable ring. Type-in light comes on.	Operator must insert the ring and enter a character on the typewriter.
60, NEEDED	System cannot locate standard input tape (logical unit 60).	Check physical tape drive containing INP for ready status. Also check that no other tapes are available to SCOPE with identical standard input label. Then, type: EQUIP, 60=HlOO, oo is the unit on which INP is mounted.
OPRELIB PARITY ERROR ON LTN xx	Parity error on specified logical unit.	Remove tape from system and/or try the job again.
PARITY ON CH xx	Transmission parity error on channel xx. Computer will STOP.	Contact customer engineer.
LIB TAP	Library tape error.	
LIB SEQ	Library tape sequence error.	
LIB EOF	Library tape end-of-file.	
LIB PAR	Library tape parity error.	
PARITY ll	Parity error on unit ll .	

## AUTOLOAD RECOVERY

Autoload recovery may be used by an operator at any time to terminate a running program abnormally.

SCOPE reads the system scratch record indicated by the ordinal set in the stop switches. SCR contains enough information to produce the recovery requested on the RUN card. If the checksum is proper, the system is intact(I). Otherwise, the system is not intact(II) .

If the recovery dump is to be on equipment other than tape, manually interrupt SCOPE during the requests for tapes to be unloaded. When the TYPE-IN light comes on after unloading all programmer tapes, enter an EQUIP statement to define 61 as non-tape.

### I. SYSTEM IS INTACT (SATELLITE MODE)

Diagnostics	Condition	Action
65=SA01	ACC is assigned to paper tape punch on the Satellite.	None
WHERE IS LAST 60 WHERE IS LAST 61 WHERE IS LAST 62	SCR has been found intact. The following messages are to locate the most recent reels of the standard units.	Type in: a 1 or 2-digit ordinal of tape assigned to requested logical unit at time of this recovery. This is obtained from the SCOPE initiation typeout: if 61 01, = MT03 is in initiation typeout, answer with 03 or 3.  If requested unit was not assigned during this job or was assigned to a unit which is not magnetic tape, a space followed by a carriage return is sufficient.
UNLOAD TAPE NO.	All standard units have been assigned. Message releases all programmer and scratch tape units in use at autoload time.	Type in: a 1 or 2-digit ordinal of a tape to be saved. This message is repeated until there is a blank and a carriage return. All tapes, 1-59, assigned within this job should be unloaded.
WHICH JOB NEXT	Recovery dump taken. INP is positioned at point where last program ceased reading.	Follow with a job sequencing statement: NEXT ENDSCOPE REPEAT Repeat an AET statement given during this job. Reassign any non-tape equipment for standard units given during this job.

**Example:**

```
70 1101, = MT 20
61 01, = MT 03
  SET STOP SWITCH 1 ON FOR SCR = MT 01
60 03, = MT 12
  0004 ,LLSLEI,
01 01, = MT 15
71 01, = MT 16
UNLOADED MT 16
60 04, = MT 14
  0005 ,DMKURN,
05 01,FTN COSY 4.00 = MT 05
69 01, = MT 12
WAITING FOR INP
60 05, = MT 12
  0006 ,EFJONE,
60 06, = MT 14
  0007 ,GHJONE,
60 07, = MT 12
  0008 ,IJJONE,
60 08, = MT 14
  0009 ,KLJONE,
60 09, = MT 16
  0010 ,MNJONE,
07 , = CP 01
69 01, = MT 12
CANNOT ASSIGN 43 = MT LOOK.

43 01, = MT 12
CANNOT ASSIGN 25 = MT LOOK.

25 01, = MT 05

65 , = SA 01

WHERE IS LAST 60
WHERE IS LAST 61
WHERE IS LAST 62

UNLOAD TAPE NO. 12

UNLOAD TAPE NO. 5

UNLOAD TAPE NO.

WHICH JOB NEXT NEXT.

0011 ,OPJONE,
```

Sequence job #10

Point of AUTOLOAD recovery

Sequence job #10 was abandoned  
and job #11 was executed.

**SYSTEM IS INTACT (NON-SATELLITE MODE)**

Diagnostics	Condition	Action
WHERE IS LAST 60 WHERE IS LAST 61 WHERE IS LAST 62 WHERE IS LAST 65	SCR has been found intact. The following messages are to locate the most recent reels of the standard units.	Type in: a 1 or 2-digit ordinal of tape assigned to requested logical unit at time of this recovery. This is obtained from the SCOPE initiation typeout: if 61 01, = MT03 is in initiation typeout, answer with 03 or 3.  If requested unit was not assigned during this job or was assigned to a unit which is not magnetic tape, a space followed by a carriage return is sufficient.
UNLOAD TAPE NO.	All standard units have been assigned. Message releases all programmer and scratch tape units in use at autoload time.	Type in: a 1 or 2-digit ordinal of a tape to be saved. This message is repeated until there is a blank and a carriage return. All tapes, 1-59, assigned within this job, should be unloaded.
WHICH JOB NEXT	Recovery dump has been taken. INP is positioned at the point where the last program ceased reading.	Follow with a job sequencing statement: NEXT ENDSCOPE REPEAT Repeat an AET statement given during this job. Reassign any non-tape equipment for standard units given during this job.



**Example:**

ENTER DATE MDY 070164

ENTER TIME HHMMSS. MARK BY JK1 111500

70 1201, = MT 20  
61 01, = MT 02  
65 01, = MT 11  
SET STOP SWITCH 1 ON FOR SCR MT = 01  
60 01, = MT 14  
101 ,CDJONE,  
102 ,EFJONE,  
103 ,GHJONE,  
104 ,IJJONE,  
105 ,KLJONE,  
106 ,MNJONE,  
07 , = CP 01  
69 01, = MT 12  
43 01, = MT 15  
RELEASED MT 12  
25 01, = MT 12  
WHERE IS LAST 60

WHERE IS LAST 61

WHERE IS LAST 62

WHERE IS LAST 65

UNLOAD TAPE NO. 12

UNLOADED MT 12  
UNLOAD TAPE NO. 15

UNLOADED MT 15  
UNLOAD TAPE NO.

WHICH JOB NEXT REPEAT.

106 ,MNJONE,  
07 , = CP 01  
69 01, = MT 04  
43 01, = MT 16  
RELEASED MT 04  
25 01, = MT 04

Normal printout during execution  
of an input reel.

Point of AUTOLOAD recovery

Sequence job 106 will be  
repeated

## II. SYSTEM IS NOT INTACT (SATELLITE MODE)

Diagnostics	Condition	Action
65 = SA01	ACC is assigned to paper tape punch on the Satellite.	None
WHERE IS LAST 70	SCR was not found intact. The following messages are used to locate the standard assignments of tape units.	Type in: A 1 or 2-digit ordinal of tape assigned to requested logical unit at time of this recovery. If unit was not tape, not assigned or equivalenced, enter space, carriage return.
WHERE IS LAST 60		
WHERE IS LAST 61		
WHERE IS LAST 62		
LEST OLD 60 BE FORGOTTEN	If the job in operation was part of a priority input, this message should be answered.	Type in: Space, carriage return if not a priority job. If priority, enter tape number of previous input tape whether or not it has been unloaded.
ENTER DATE MDY		Enter 6 digits, MMDDYY.
ENTER TIME HHMMSS. MARK BY JK1		Enter time and set jump key 1 to mark the second when clock should be read.
UNLOAD TAPE NO.	All standard units have been assigned. Message releases all programmer and scratch tape units in use at autoload time.	Type in: A 1 or 2-digit ordinal of tape to be saved. This message is repeated until there is a blank and a carriage return. All tapes, 1-59, assigned within this job should be unloaded.
WHICH JOB NEXT	Recovery dump has been taken. INP is positioned at the point where the last program ceased reading.	Redefine any non-tape standard units. Re-enter any AET statements previously given which should remain in effect. Follow with a job sequencing statement: NEXT, REPEAT, ENDScope. Terminate input by period, carriage return, when all messages have been entered. Messages may be in any order.

**Example:**

65 , = SA 01  
WHERE IS LAST 7020

WHERE IS LAST 60

WHERE IS LAST 613

WHERE IS LAST 62

LEST OLD 60 BE FORGOTTEN15

ENTER DATE MDY 070164

ENTER TIME HHMMSS. MARK BY JK1 040000

UNLOAD TAPE NO. 5

UNLOAD TAPE NO. 14

UNLOAD TAPE NO.

WHICH JOB NEXT NEXT.

JOB ABANDONED  
CANNOT ASSIGN 61 = MT LOOK.

61 0101, = MT 05  
0015 ,GHJONE,  
0016 ,IJJONE,  
0017 ,KLJONE,  
0018 ,MNJONE,  
07 , = CP 01  
69 01, = MT 16  
43 01, = MT 03  
25 01, = MT 16NEXT.

**SYSTEM IS NOT INTACT (NON-SATELLITE MODE)**

Diagnostics	Condition	Action		
WHERE IS LAST 70 WHERE IS LAST 60 WHERE IS LAST 61 WHERE IS LAST 62 WHERE IS LAST 65	SCR was not found intact. The following messages are used to locate the standard assignments of tape units.	Type in: A 1 or 2-digit ordinal of tape assigned to requested logical unit at time of this recovery. If unit was not tape, not assigned or equivalenced, enter space, carriage return.		
ENTER DATE MDY			Enter 6 digits, MMDDYY.	
ENTER TIME HHMMSS. MARK BY JK1			Enter time and set jump key 1 to mark the second when clock should be read.	
UNLOAD TAPE NO.			All standard units have been assigned. Message releases all programmer and scratch tape units in use at autoload time.	Type in: A 1 or 2-digit ordinal of a tape to be saved. This message is repeated until there is a blank and a carriage return. All tapes, 1-59, assigned within this job should be unloaded.
WHICH JOB NEXT			Recovery dump has been taken. INP is positioned at point where last program ceased reading.	Redefine non-tape standard units. Re-enter AET or PAUSE statements previously given which should remain in effect. Follow with a job sequencing statement: NEXT, REPEAT, ENDScope. Terminate input by period, carriage return, when all messages have been entered. Messages may be in any order.

**Example:**

```
ENTER DATE MDY 070164

ENTER TIME HHMMSS. MARK BY JK1 114500

70 1201, = MT 20
61 01, = MT 02
65 01, = MT 04
  SET STOP SWITCH 1 ON FOR SCR = MT 01
60 01, = MT 14
  101 ,CDJONE,
  102 ,EFJONE,
  103 ,GHJONE,
  104 ,IJJONE,
  105 ,KLJONE,
  106 ,MNJONE,
07 , = CP 01
69 01, = MT 11
43 01, = MT 12
RELEASED MT 11
25 01, = MT 11
```

Normal printout during execution  
of an input reel.

Point of AUTOLOAD recovery

WHERE IS LAST 7020

WHERE IS LAST 6014

WHERE IS LAST 612

WHERE IS LAST 62

WHERE IS LAST 654

ENTER DATE MDY 070164

ENTER TIME HHMMSS. MARK BY JK1 120300

UNLOAD TAPE NO. 12

UNLOADED MT 12  
UNLOAD TAPE NO. 11

UNLOADED MT 11  
UNLOAD TAPE NO.

WHICH JOB NEXT NEXT,107

```
JOB ABANDONED
107 ,ABMILL,
108 ,MNJONE,
109 ,FINISH,
60 02, NEEDED
```

# INDEX

---

- Abnormal termination 2-26
- Absolute binary records 7-4
- Acceptable requests 3-3
- ACC, Accounting information, 1-5
- Accounting information 1-4, 2-2
- AET, entry description 2-22
  - format A-1
  - listing C-20
  - statement 2-21
- ALDAP 2-16
  - options 2-16
- ALGO 2-15
- ALGOL 2-15
- Alter AET entry 2-22
- Areas of recovery dump 2-26
- Assignment declaration 2-6
  - problems 1-10
- Autoload recovery C-32
- Auxiliary libraries 1-3, 1-4
- Available equipment table A-1
  
- BANK statement 5-3
- Bank assignment 5-10, 6-4
- BCD records 7-4
- BCT card 5-21
- Binary cards 5-16
  - BCT 5-21
  - BRT 5-27
  - EPT 5-20
  - EXT 5-24
  - IDC 5-19
  - LAT 5-24
  - LCC 5-29
  - OCC 5-30
  - RBD 5-22
  - TRA 5-28
- Binary card format 5-18
- Blank labels 1-5
- BOUND/UNBOUND request 3-18, B-8
- BRT card 5-27
  
- Changing library tape 7-19
- Charge number 2-2
- Checksum 5-18
- Clock interrupt 3-19
- COBOL 2-14
  - options 2-14
- Comment card 7-8
- Control statements
  - DELETE 7-14
  - DIR 7-6
  - EDIT 7-12
  - END 7-6
  - EOF 7-6
  - EXTRACT 7-17
  - FINISH 7-7
  - INSERT 7-15
  - LIST 7-11
  - PRELIB 7-7, 7-19
  - PREPARE 7-16
  - REPEAT 7-8
  - REPLACE 7-16
  - SCOPE 1-2, 2-2
  - UNIT 7-9
- COMPASS 2-12
  - options 2-12
- Compiled TRA 5-2
- CORE request 3-24, B-10
- CORRECT statement 5-6
- Correcting subprograms 5-13
  
- Data fields 5-15
- DATE request 3-21, B-8
- Debugging aids 1-1, 4-1
  - parameters 2-18
- Deck examples 2-30
- Deck structure 6-4
- Declarations
  - assignment 2-6
  - density 2-5, 3-7
  - direction 3-7
  - disposition 2-6

## Declarations

- equivalence 2-7
- hardware 2-3, 2-4
- labeled tape 2-9
- logical unit number 2-10
- unlabeled tape 2-8
- usage 2-5, 3-6, B-4

DELETE control statement 7-14

Density declarations 2-5, 3-6

- types 2-5, 3-6, B-4

## Diagnostics

- loader C-24

- LOADMAIN 6-12

- on OCM C-30

- on OUT C-21

- PRELIB C-28

- recovery dump C-22

- SNAP/TRACE C-23

DIR control statement 7-6

Disposition of tapes 1-3

- types 2-6

Dump parameters 4-1, 4-3

- recovery 4-5

- SNAP 4-1

- TRACE 4-3

EDIT control statement 7-12

Editing library tape 7-12

END control statement 7-6

End-of-file card 2-28

ENDLIB\* statement 7-5

ENDM statement 7-5

END REEL statement 2-2, 2-23

ENDSCOPE statement 1-2, 2-2, 2-24

ENDSYS\* statement 7-5

Entry point name statement 2-3, 2-12, 2-18

EOF control statement 7-6

EPT card 5-20

Equipment assignment 1-5

- declarations 2-3

EQUIP statement 1-5, 2-3

Equivalence declarations 2-7

ERASE request 3-5, B-2

Executing object programs 2-26

EXIT request 3-23, B-8

EXT card 5-24

External interrupts 3-14

EXTRACT control statement 7-17

FILE statement 2-2

FILE END statement 2-2

FINISH control statement 7-7

Force bank 5-5

Format of overlay tapes 6-1

FORTRAN 2-13

- call overlays 6-14

- options 2-13

FREE request 3-20, B-8

General macros B-1

Hardware declarations 2-3

- types 2-4, A-1

Held requests and interrupts 3-13

HERESAQ request 3-24, B-8

ICM - see input comment 1-5

- messages C-15

IDC card 5-19

Indirect interrupt select 3-15

Initiation timeout C-19

INP, standard input, 1-5

Input comment 1-3

Input/output control 1-1

- requests 3-1, B-3

INSERT control statement 7-15

Internal interrupt control 3-14

Interrupts

- clock 3-18

- external 3-14

- internal 3-14

- types 3-15

Job sequence number 2-1

- stack 1-2

- time limit 2-2

JOB statement 1-2, 2-2

LAT card 5-24

Label

- blank 1-5

- reading 1-8

- request 1-5, B-6

- skipping 1-9

- tape 2-9

- tape declaration 2-9

- writing 1-7

LCC card 5-29  
 LIB, SCOPE library 1-5  
 Library 1-4  
   maintenance 7-1  
   preparation 1-1, 7-1  
   programs 2-12  
   request 3-21, B-9  
 LIBRARY statement 2-21  
 LIMIT request 3-19, B-8  
 List AET 2-22, C-17  
 LIST control statement 7-11  
 List library contents 7-11, 7-19  
 LOAD statement 2-25  
 Loader, position 1-2  
 Loader calls 5-7  
   cards 5-2, 5-17  
   control statements 6-1  
   diagnostics C-24  
   names 5-2  
   operations 5-1, 5-9  
   parameters 5-8  
   request 3-22, 5-7, B-9  
 Loader statements  
   OVERLAY 6-2  
   MAIN 6-2  
   SEGMENT 6-3  
 Loading from standard input 2-24  
   object programs 2-24  
   overlays 6-13  
 Load-and-go unit 1-3, 1-4, 2-1  
 LOADMAIN diagnostics 6-12  
   statement 6-1  
 Logical units 1-2  
   master 2-7  
   number declaration 2-10  
 Lower bound 3-18  
  
 Macro calling sequences B-1  
   definitions 7-5, B-1  
 MAIN loader statement 6-2  
 Map, memory 4-5, C-6  
 MARKEF request 3-5, B-2  
 Master logical unit 2-7  
 Master unit, status 3-7  
 Memory map 4-5  
   map indicator 2-26  
   request 3-23, B-10  
  
 Messages and diagnostics C-1  
   on ACC C-20  
   on ICM C-15  
   on OCM C-18  
   on OUT C-1  
   on PUN C-21  
 Minimum configuration 1-5  
 MODE request 1-5, 3-6, B-5  
 Monitor, SCOPE 1-1  
  
 Normal termination 2-26  
  
 Object programs  
   executing 2-26  
   loading 2-25  
 OCC card 5-30  
 OCM, output comment, 1-5  
 Octal correction cards 2-18, 5-13, C-14  
 OUT, standard output, 105  
 Output comment 1-4  
 Output limit 2-26  
 OVERLAY  
   loading 6-13  
   loading statement 6-2  
   processing 1-1, 6-1, 6-5, 6-9  
   processing, FORTRAN 6-14  
   program, execution 6-9  
   tape 6-1  
  
 Parameters, dump 4-1, 4-3  
   loader 5-8  
   SCOPE 2-18  
   statement 1-2  
 PRELIB control statement 7-7, 7-19  
   diagnostics C-28  
   processing 7-1  
 PREPARE control statement 7-16  
 Prepare new library tape 7-16  
 Processing prepared overlay tape  
   assignment 5-10  
   extension area 5-15  
   termination 2-27  
 Programmer identification 2-2  
   requests 1-2, 1-3, 3-1  
   units 1-3, 2-1  
 PUN, standard punch output 1-3



RBD card 5-22  
READ/WRITE request 3-4, B-2  
RDLABEL/WRLABEL 3-4, B-2  
Records  
  absolute binary 7-4  
  BCD 7-4  
  relocatable binary 7-3  
  types 7-3  
Recovery dump 4-5, C-7  
  dump, areas 2-27  
  dump diagnostics C-22  
Recovery indicator 2-27  
Release code 3-6  
RELEASE request 3-6, B-4  
Relocatable binary records 7-3  
Relocation factors 5-14  
REOT/WEOT request 3-5, B-2  
REPEAT control statement 7-8  
Repeated library records 7-8  
REPLACE control statement 7-16  
Requests  
  BOUND/UNBOUND 3-18, B-8  
  CORE 3-24, B-10  
  DATE 3-21, B-8  
  ERASE 3-5, B-2  
  EXIT 3-23, B-8  
  FREE 3-20, B-8  
  HERESAQ 3-24, B-8  
  input/output 3-1, B-2  
  LABEL 1-5, 3-10, B-6  
  LIBRARY 3-21, B-9  
  LIMIT 3-19, B-8  
  LOADER 3-22, 5-7, B-9  
  MARKEF 3-5, B-2  
  MEMORY 3-23, B-9  
  MODE 1-5, 3-6, B-5  
  programmer 1-2, 3-1  
  RDLABEL/WRLABEL 3-4, B-2  
  READ/WRITE 3-9, B-2  
  RELEASE 3-6, B-4  
  REOT/WEOT 3-5, B-2  
  REWIND 3-5, B-2  
  SAVE 1-5, 3-11, B-6  
  SELECT/REMOVE 3-15, B-7  
  SKIP 3-5, B-2  
  special 3-21, B-9  
  stacking 3-12  
  STATUS 3-7, B-5  
  TIME 3-20, B-8  
  UNLOAD 3-5, B-4  
  UNSAVE 3-12, B-6  
  Restrictions on EQUIP 2-6  
  REWIND request 3-5, B-2  
  RUN statement 2-3, 2-26  
  Run time limit 2-26  
  SAVE request 1-5, 3-11, B-6  
  Satellites 1-6, C-32  
    communication 1-1  
SCOPE  
  call codes B-1  
  control statements 2-1  
  initiation type-out C-19  
  library 1-4  
  monitor 1-1  
  parameters 2-18  
  resident 1-2  
  routines 1-2  
  SCR - see system scratch record, 1-5  
  Scratch units 1-3, 2-1  
  SEGMENT loader statement 6-3  
  SELECT/REMOVE request 3-15, B-7  
  SEQUENCE statement 2-1  
  SKIP request 3-5, B-2  
  SNAP cards 2-19  
    dump 4-1, C-11  
  SNAP/TRACE diagnostics C-23  
  Source deck structure 2-16  
  Special requests 3-20, B-8  
  Stack, job 1-2  
  Stacking requests 3-12  
  Standard input 1-3  
  Standard output 1-3  
  Standard punch output 1-3  
  Statements  
    AET 2-22  
    BANK 5-3  
    CORRECT 5-6  
    DELETE control 7-14  
    DIR control 7-6  
    EDIT control 7-12  
    END control 7-6  
    ENDLIB\* 7-5  
    ENDM 7-5  
    END REEL 2-2, 2-23  
    ENDSCOPE 1-2, 2-2, 2-24  
    ENDSYS\* 7-5

## Statements

- entry point name 2-3, 2-12, 2-18
- EOF control 7-6
- EQUIP 1-5, 2-3
- EXTRACT control 7-17
- INSERT control 7-15
- FILE 2-2
- FILE END 2-2
- FINISH control 7-7
- JOB 1-2, 2-2
- LIST control 7-11
- LOAD 2-24
- LOADER control 6-1
- LOADMAIN 6-1, 6-11
- MAIN loader 6-2
- OVERLAY loader 6-2
- PRELIB control 7-7, 7-19
- PREPARE control 7-16
- REPEAT control 7-8
- REPLACE control 7-16
- RUN 2-3, 2-26
- SCOPE control 1-2
- SEGMENT loader 6-3
- SEQUENCE 2-1
- UNIT control 7-9
  - SCOPE control 1-2
- Statement parameters 1-2
- STATUS request 3-7, B-5
- Status of master unit 3-7
- Storage allocation 5-10
- Storage diagram 5-12, 6-4
- System scratch record 1-4
- System units 1-3

## Tape control requests 3-5

- disposition 1-3
- labels 2-9
- labeling 1-6
- units - see logical units

## Time limit 3-19

- job 2-2
- TIME request 3-20, B-8
- TRA card 5-28
- TRACE cards 2-19
  - dump 4-3, C-11

## Typewriter character codes C-17

## Units, logical 1-2

- programmer 1-3
- scratch 1-3
- system 1-3
- UNIT control statement 7-9
- Unlabeled tape 2-8
- Unlabeled tape declaration 2-8
- UNLOAD request 3-6, B-4
- UNSAVE 3-12, B-6
- Upper bound 3-18
- Usage declarations 2-5, 3-6, B-4
  - types 2-5, 3-6, B-4
- User control card 7-5

## Write assignment 1-7

**CONTROL DATA**

C O R P O R A T I O N

**COMMENT AND EVALUATION SHEET**

3600 Computer System

SCOPE Reference Manual

Pub. No. 60053300, Rev. B

June, 1965

YOUR EVALUATION OF THIS MANUAL WILL BE WELCOMED BY CONTROL DATA CORPORATION. ANY ERRORS, SUGGESTED ADDITIONS OR DELETIONS, OR GENERAL COMMENTS MAY BE MADE BELOW. PLEASE INCLUDE PAGE NUMBER REFERENCE.

**FROM** NAME : \_\_\_\_\_

**BUSINESS ADDRESS :** \_\_\_\_\_

**NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.**

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS  
 PERMIT NO. 8241  
 MINNEAPOLIS, MINN.

**BUSINESS REPLY MAIL**  
 NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

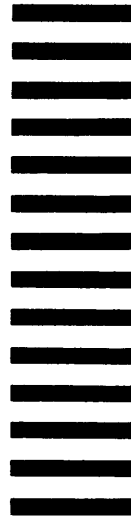
POSTAGE WILL BE PAID BY

**CONTROL DATA CORPORATION**

*Documentation Department*

3145 PORTER DRIVE

PALO ALTO, CALIFORNIA



FOLD

FOLD

STAPLE

STAPLE



▶ ▶ CUT OUT FOR USE AS LOOSE-LEAF BINDER TITLE TAB

**CONTROL DATA**  
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN, 55440  
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD