

UNIVAC[®] 1107

TECHNICAL BULLETIN

EXEC I

UNIVAC 1107 EXECUTIVE SYSTEM

TABLE OF CONTENTS

	Page
TABLE OF CONTENTS.....	i to iii
I. INTRODUCTION.....	1
A. Schedule Maintenance.....	1
B. Selection.....	1
C. Facility.....	1
D. Loading.....	1
E. Interrupt.....	2
F. Input/Output.....	2
G. Switching.....	2
H. Communication.....	3
I. Logging.....	3
J. Dumping.....	3
K. Termination.....	3
II. SCHEDULE MAINTENANCE.....	5
A. Submission of Job Request.....	5
B. Job Request Format.....	6
C. General Schedule Information.....	17
D. Schedule by Paper Tape.....	18
E. Deletion of Job Requests.....	21
III. SELECTION.....	23
A. Operation.....	23
B. Method of Selection.....	23
C. Facility Check.....	26
D. Selection Interruption.....	27
E. EXEC Message Relating to Job Selection.....	27
IV. FACILITY ASSIGNMENT.....	29
A. Facility Description (FAC Card).....	30
B. Changes in Facility Requirements.....	33
C. Allocation of Facilities.....	35
D. Operator Notification.....	37
E. Release of Facilities.....	37
F. EXEC Rules for Return of Facilities to Available Status.....	38
V. LOADING.....	41
A. Location of Job Program.....	41
B. Location of Subroutines for Complex Programs....	46
C. Operation and Main Function of Loader.....	47
D. Initiation of Job Programs.....	70

TABLE OF CONTENTS (cont.)

	Page
VI. INTERRUPT.....	73
VII. INPUT/OUTPUT.....	77
A. Submission of Requests.....	78
B. I/O Execution Packet: General.....	79
C. I/O Execution Packet: Symbolic Form.....	87
D. I/O Functions.....	91
E. I/O Errors and Recovery.....	108
VIII. SWITCHING.....	117
A. The Dispatcher.....	117
B. Program Release of Control.....	119
C. The Switch Lists.....	120
D. Storing and Restoring Film Memeory.....	121
IX. COMMUNICATION.....	123
A. Communication Requests.....	123
B. Requests Parameter.....	123
C. Execution Packet.....	124
D. Communication Functions.....	127
E. Communication Conventions.....	129
X. LOGGING.....	133
A. Running Time Log.....	134
B. Job Initiation Log.....	135
C. Job Termination Log.....	137
D. Input/Output Error Log.....	139
E. Trouble Dump.....	142
F. Date Blocks on Log Medium.....	145
XI. DUMPING FUNCTION.....	147
A. Automatic Dump.....	147
B. Program Requested Dump.....	147
XII. TERMINATION.....	149
A. Normal Job Termination.....	149
B. Abnormal Job Termination.....	149
C. Termination Specified by the Operator.....	150
D. Manual Termination of Program by Operator.....	150
E. Temporary Interruption of a Program.....	151
F. Termination Timeouts.....	152

TABLE OF CONTENTS (cont.)

Page

XIII. EXECUTIVE PROCEDURES.....	155
A. Initialization.....	155
B. Unsolicited Messages.....	157
C. Execution of Rush Jobs.....	158
D. Facility Transfer Function.....	169
E. Assembly (Compilation) and Testing of Programs.....	180
F. Job Program Libraries.....	181
G. Rerun Function.....	187
H. Independent Operation of Jobs.....	190
XIV. DATE AND TIME.....	193
A. Date.....	193
B. Time.....	193
APPENDIX A. IBM BCD Characters and Tape Codes (Octal).....	195
APPENDIX B. Job Request Error Codes.....	197
APPENDIX C. Loading Error Codes.....	199
APPENDIX D. Termination Codes.....	201
APPENDIX E. EXEC I/O Function List.....	203
APPENDIX F. I/O Packet Status Codes.....	207
APPENDIX G. Packet Formats and Calling Sequences.....	211
APPENDIX H. Internal Transfer Codes.....	219
APPENDIX I. 1107 Subsystem Codes.....	221
APPENDIX J. EXEC I/O Request for UNISERVO IIIA Dual Channel Subsystem.....	223
APPENDIX K. Error Table.....	225

I. INTRODUCTION

EXEC, the 1107 Executive System, is a program designed to provide for automatic processing of a scheduled set of computer runs, allocation of memory and peripheral units, input/output operations, concurrent processing of 1107 programs.

EXEC is intended to facilitate efficient use of the UNIVAC® 1107 Thin Film Memory Computer by providing the means for automatically processing a scheduled set of jobs with a minimum of operator intervention. Jobs may be processed concurrently or serially as specified in the externally prepared Job Requests.

EXEC occupies core locations 0-7777 and 130000-137777 octal for a 32K or 49K core memory or 0-7777 and 170000-177777 octal for a 65K core memory. Drum locations 0-105000 octal are reserved for EXEC segment storage.

To accomplish its intended purpose. EXEC must perform a varied number of functions. These include:

A. Schedule Maintenance

The acceptance of Job Requests from an external medium and the inclusion of these requests in a Job Request Schedule. EXEC will reference the Job Request Schedule to determine the next job to be initiated. Previously submitted requests may be deleted.

B. Selection

The use of information contained in the Job Request Schedule to select the next job to be initiated. Selection is based on the priority and precedence assigned to the job, the sequence relationship of this job to other jobs with the same priority and precedence, and the availability of facilities required by the job.

C. Facility Assignment

The assignment of memory and external facilities to meet the requirements which are defined symbolically in a job program selected for initiation. EXEC maintains a list of all allocatable facilities which is updated to reflect assignment of facilities to newly initiated programs and to reflect release of facilities by programs during, or at termination of, a run.

D. Loading

The transfer of a job program to be initiated from the storage medium to the absolute operational facilities

assigned to the program. Programs which are in relocatable form (ROC) are transferred in their entirety to the assigned operational facilities. All necessary modifications are made during this transfer to make the program operationally compatible with the assigned absolute facilities. Each program is assigned an internal identification upon loading and is then initiated. All programs are loaded by the EXEC Loader, the 1107 Relative Load routine within the EXEC.

E. Interrupt

The act of providing to the operating program the entrances to subroutines which will handle the error interrupts. Upon occurrence of an error interrupt, control is transferred automatically to one of the fixed core-memory addresses 192-199. EXEC provides jump instructions for these locations. These instructions in turn reference subroutines which will attempt to recover from these errors. Attempts to write into a locked out area of memory results in program termination when operating under EXEC control. Recovery routines are permitted for illegal operation code, trace mode, characteristic overflow, characteristic underflow, and divide fault interrupts. All I/O and other interrupts are handled by other parts of the Executive System.

F. Input/Output

The acceptance, listing, and processing of all requests for I/O functions from the operating programs. This function of the Executive System makes possible the concurrent operation of several programs using the same I/O channels without the danger of one program interfering with another program's I/O functions. Requests for I/O operations are submitted to EXEC in the form of a parameter specifying the location of an "execution packet" which defines the function to be performed. An attempt is made to recover from I/O errors whenever feasible.

G. Switching

Providing for the transfer of operational control among two or more independent programs being operated concurrently. This function makes possible the operation of compute-limited programs concurrently with I/O-limited programs. Operational control is transferred to a compute-limited program whenever the I/O-limited programs must wait, pending completion of requested I/O functions. This function of EXEC allows an installation to make maximum use of its total computer facility.

H. Communications

Providing for all communication between the operating programs and the computer operator and between the Executive System and the computer operator. These communications take place via the computer keyboard and the on-line typewriter. This function includes the interpretation of all keyboard inputs addressed to the Executive System and the transfer of control to the section of EXEC to which the input pertains.

I. Logging

The recording of the approximate internal processing times utilized by each program operating concurrently as well as the unused time during which no program can operate pending completion of requested functions. This record will assist the installation scheduler in determining which programs to run in parallel with each other. To facilitate the detection of infinite loops, provision is made to notify the operator of programs which utilize more time than is estimated for them.

J. Dumping

The facility to obtain printable dumps of the contents of areas of film or core memory in case unexpected errors cause premature termination of supposedly debugged programs. The dumps are recorded on tape for later printing on the High-Speed Printer.

K. Termination

The normal or abnormal termination of an operating program and the return of its assigned facilities to an "available" status. Termination may be initiated by EXEC, by the job program, or by the operator.

Each of these functions of the Executive System will be considered in detail in the following pages. It is assumed that the reader is familiar with the pertinent facts concerning the hardware and also with SLEUTH, the 1107 Assembly System. The format of the Relative Object Code (ROC) is discussed in the manual on CLAMP, the 1107 Relative Load Routine. Reference should be made to both the SLEUTH and CLAMP manuals and to the interrelationships of these systems with EXEC. A functional diagram of EXEC appears as Figure 1.

1107 EXECUTIVE SYSTEM
FUNCTIONAL DIAGRAM

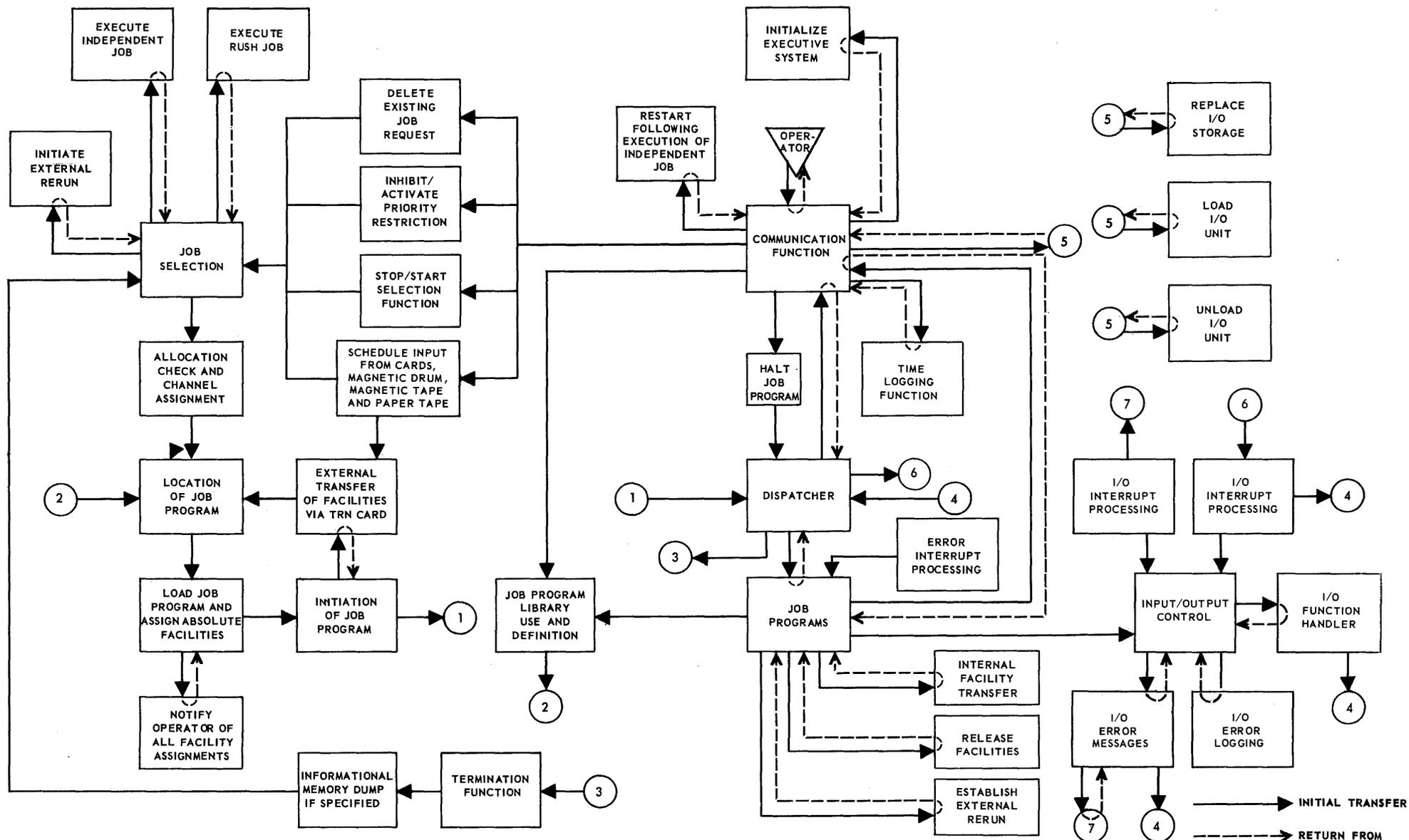


FIGURE 1

II. SCHEDULE MAINTENANCE

Before a job program can be selected as a candidate for loading and initiation, a Job Request must be submitted to the Executive System. The Job Request consists of a series of input parameters to EXEC specifying the program to be initiated, its location, modifications to be made while loading, and all other pertinent information concerning the program.

A. Submission of Job Requests.

1. Operator Initiation of Schedule

Before job requests can be submitted, the operator must notify the Executive System of the location of the job requests. This is accomplished via a type-in of the form.

SCH△Ccc△Uuu

This form is applicable to job requests which are contained on all types of peripheral units except magnetic drum and the computer-operator console. In this type-in:

SCH identifies the type-in as a request to EXEC to read in job requests.

Ccc△Uuu specifies the peripheral unit containing the job requests; cc is the absolute channel number and uu the absolute unit number.

Units which may be specified are a card reader, a magnetic tape unit, or a paper tape reader. The Job Requests are read by EXEC from the specified input device, checked for legality, and stored in the schedule of Job Requests. The existing Job Request Schedule is updated to contain only uncompleted Job Requests.

If several schedules of job requests are to be stacked on a magnetic tape, each schedule must be identified by a unique alphanumeric identifier in columns 41 and 42 of the start card image. When it is desired to select jobs from this medium, the identifier must be given as part of the SCH message as follows:

SCH△Ccc△Uuu△n2

where n2 is the alphanumeric identifier.

If the specified input unit is a magnetic tape and is currently assigned to a job program or to EXEC, or contains a program library, the operator is notified by a type-out and the schedule maintenance function of EXEC is terminated.¹ If the input unit has already been placed in a "reserved" or "available" status by EXEC (for use during program execution), the job requests are read and the unit is restored to its original status. EXEC takes over the assignment of the card reader or paper tape reader long enough to read in the schedule.

¹ Program libraries are discussed in Section XIII.

The EXEC reserves 4000 magnetic drum locations for storage of job requests. The size of a request is usually from 15 to 40 words; hence from 100 to 260 job requests can be stored on drum.

2. Program Initiation of Schedule

A group of job requests stored on either magnetic tape or magnetic drum may be transferred to EXEC via the internal transfer system (see Section XIII.) The job request must be in external format as described in UP2581. If the schedule is internally transferred, any console error messages will be preceded by the letters INT. If more than one schedule of job requests is stored on the magnetic tape, the alphanumeric identifier in column 41 and 42 of the start card image must be stored in Fielddata notation in S1 and S2 of the parameter word of the transfer packet prior to performing the internal transfer of facility.

B. Job Request Format

A job request consists of a minimum of two cards.² These are the priority card (PTY) and the facility description card (FAC). These two cards must be present for all jobs to be operated under the Executive control, except for reruns and independent jobs.

Job requests for initiating an independent job or a job program from a previously established rerun point require only the PTY card.

All cards of a job request must be grouped together. Two job requests must not have the same identity. The order of card types in a job request may be random, except that the PTY card must be first and the FAC card second. Cards must, however, be grouped by type whenever more than one card of a type is present.

Certain other cards may be present. These include the table length assignment card (TAL), the transfer card (TRN), and the parameter card (PMn). Their presence is optional and they are used if specific table lengths are to be increased, if facilities are to be transferred to succeeding jobs, and/or if the job program requires starting parameters which are unique to each run.

Each schedule deck must be preceded by a start card, the format of which is defined by LION. Columns 13-20 of this card must contain the word SCHEDULE. If converted to tape or magnetic drum, the schedule must be in LION external format and be headed by a label block with the Fielddata characters SCHEDULE in words 2 and 3 of this block.

²The term "card" is used here and throughout this manual to indicate a unit record which could take the form of punched cards, a magnetic tape record, or a paper tape record. Actually a single record may consist of more than one punched card. Each record, or card, may contain a maximum of 80 characters.

1. Priority card (PTY): The PTY card is the basic card for every job request and consists of 14 ordered fields separated by commas. Two consecutive commas are necessary to show the omission of a field except when trailing fields are omitted. All spaces and blanks are ignored.

The fields and the information contained therein are as follows:

- JOB REQUEST ID: From one to six alphanumeric characters which identify the job request.
- CARD TYPE: The characters "PTY" to identify the card as a priority card.
- CARD COUNT: Count of number of cards in the job request written in decimal notation.
- RUSH: This field may contain "R", "P" or "S", where "R" denotes a rush job, S places the new job request in immediate suspension and P indicates a permanent job.
- PROGRAM NAME³: From one to twelve alphanumeric characters which identify the job program or six characters that identify the rerun dump.
- RUN NOW: When set with "I", the program will be run immediately upon loading without waiting for operator initiation.
- PROGRAM MEDIUM: This field defines the type of input medium on which the job program resides. It contains:
- a) The symbol "T" to denote UNISERVO* IIIA, or
 - b) the symbol "A" to denote UNISERVO IIA, or
 - c) the symbol "C" to denote UNISERVO IIIC, or
 - d) The symbol "*" to denote that the MEDIUM NAME field which follows contains an absolute assignment.
- MEDIUM NAME: This field contains the name or location of the medium containing the job program. It contains either:
- a) the symbol "CccΔUuu" where cc and uu specify the absolute channel and unit location of the job program, or
 - b) the symbol "*t = p/rr" where * defines the tape symbol t as denoting a library of programs. This designation is optional. The tape symbol t is the alphanumeric

³This is the tag which appears in the PRO line of a SLEUTH source program. It is expanded to 12 characters by the Assembly System with the right-most 6 characters filled with spaces.

name (one to six characters) of the tape containing the job program. This is the only subfield which must be present if this form of the MEDIUM NAME field is used.

The = symbol equates the tape symbol to the program symbol p. The latter is the alphanumeric name (one to six characters) used in the job program to refer to a tape. The presence of the symbol p in this field causes all job program references to this symbol to be replaced at load time by the absolute channel and unit assignment of the tape containing the job program. The logical channel associated with this tape is defined with an asterisk on the FAC Card (see below).

rr consists of two decimal digits representing the relative location of this job program on the program library tape named in the t subfield. This subfield is used only when the asterisk (*) subfield is present.

The MEDIUM NAME field is further discussed in Section V.

RUN TYPE:

This field may contain

- a) the symbol "S" to indicate that this job must run serially.
- b) the symbol "P" to indicate that this job may run concurrently with other jobs.
- c) the symbol "E" to indicate that this job is an external rerun (See Section XIII).

- d) the symbol "I" to indicate that this program must run independently.
- e) the symbol "R" to indicate that this job is a real-time program.

PRIORITY:

From 1 to 5 characters containing the sequence, priority, and precedence assignments for the job request. The first two decimal digits are sequence numbers which specify the order of execution of jobs with identical priorities and precedences other than Z63. The third character in this field is the alphanumeric priority, with "A" signifying the highest priority. The last two characters are decimal digits representing the precedence value. The minimum requirement for this field is the alphabetic priority. If the precedence is omitted, it is given a value of 63. For this case the priority must not be Z. (Z 63 is reserved for permanent jobs.) The precedence values range from 0 to 63, while the sequence values may vary from 1 to 63.

MIX TYPE:

This field assigns the job to either the compute-limited class or to the I/O limited class of programs. It is used as a basis for distributing control to the operating job programs (see Section VIII). This field may contain.

- a) the symbol "P" to denote an I/O limited program, or
- b) the symbol "C" to denote a compute-limited program.

ESTIMATED TIME:

Up to four decimal digits giving an estimate, to the nearest minute, of the central computer time needed for the operation of the program. This estimate should be above the maximum and will be used to detect an infinite loop in a program.

LIBRARY MEDIUM:

This field defines the type of input tape which contains the subroutines to be included with

JOB REQUEST ID	CARD TYPE	CARD COUNT	RUSH	PROGRAM NAME	RUN NOW	PROGRAM MEDIUM	MEDIUM NAME	RUN TYPE	PRIORITY	MIX TYPE	EST. TIME	LIBRARY MEDIUM	LIBRARY NAME
----------------------	--------------	---------------	------	-----------------	------------	-------------------	----------------	-------------	----------	-------------	--------------	-------------------	-----------------

965004	, PTY ,	4	, R ,	RRU-PAYROL-3,	I ,	T	, *PROG=ABLE/04,	S ,	01A23	, P ,	1254 ,	A ,	STRESS.
--------	---------	---	-------	---------------	-----	---	------------------	-----	-------	-------	--------	-----	---------

965006	, PTY ,	5	, R ,	TIME-FILE	,	A	, BF3	, P ,	A24	, C ,	2 .		
--------	---------	---	-------	-----------	---	---	-------	-------	-----	-------	-----	--	--

253	, PTY ,	6	, ,	DUMP	,	T	, PT2	, P ,	B6	, C ,	1 .		
-----	---------	---	-----	------	---	---	-------	-------	----	-------	-----	--	--

Three PTY Cards are shown. In the first of these, the job program RRU-PAYROL-3 resides on a UNISERVO IIIA tape unit (denoted by "T" in the PROGRAM MEDIUM Field). The tape containing the job program is named PROG and is a program library tape (as denoted by the "*"). The job program is the fourth program on the tape PROG. All references within the job program to the tape ABLE are replaced at load time by the absolute channel and unit assignment given to the tape PROG. This program is a rush job which may temporarily usurp the core area used by one or more operating programs. It is I/O limited and utilizes subroutines stored on a tape named STRESS which is mounted on a UNISERVO IIA tape unit.

FIGURE 2 PTY CARD

JOB REQUEST ID	CARD TYPE	FACILITY DESCRIPTIONS
965004	FAC	DB/3 , IB/2 , MD/1500 , MA1/2 , MT1/4 , *MT2/3 ,
MT/2: PAYROL: VTORY , JP/3 .		

The facility requirements described by this card are as follows:

<u>Facility</u>	<u>Number of Units</u>
Instruction Bank (IB)	2 core blocks (2048 words in each; 4096 total)
Data Bank (DB)	3 core blocks (2048 words in each; 6144 total)
Magnetic Drum (MD)	1500 registers
UNISERVO IIA tape (MA)	2 tape units on logical channel 1
UNISERVO IIIA tape (MT)	4 tape units on logical channel 1
UNISERVO IIIA tape (*MT)	3 tape units on logical channel 2
UNISERVO IIIA tape (MT)	2 tape units on one or more channels (The units PAYROL and VTORY are not to be assigned for this run of the program (see text)).
Selective Jump Switches (JP)	3 switches

The asterisk indicates that the tape containing the job program is associated with logical channel 2 and, if not already assigned, will be assigned to the same physical channel to which logical channel 2 is assigned.

FIGURE 3: FAC CARD

the job program at load time (see manual on CLAMP). It may contain

- a) the symbol "T" to denote UNISERVO IIIA, or
- b) the symbol "A" to denote UNISERVO IIA, or
- c) the symbol "C" to denote IIIC, or
- d) the symbol "*" to denote that the LIBRARY NAME field which follows contains an absolute assignment.
- e) the symbol Z to denote that the library medium is the same as the program medium.

LIBRARY NAME: This field contains the name of the library tape referred to above (see manual on 1107 LIBRARIAN).

The last field is followed by a period to indicate end-of-record. The PROGRAM NAME field may contain a hyphen ("-") character except as the first character in that field. When the request is for an external rerun, the PROGRAM NAME field identifies the rerun dump (of which more than one may exist). Figure 2 contains three PTY cards for three different Job Requests. These cards follow the form described above.

2. Facility Description Card (FAC)

This card must be present for all jobs which are to be executed under EXEC control. The card contains a JOB REQUEST ID field which is identical to the one described above for the PTY card, a CARD TYPE field with the symbol "FAC" to identify the card, and a number of FACILITY DESCRIPTION fields. These latter fields will be described in detail in Section IV on Facility Assignment. All required facilities other than those that are being transferred internally to the job program from another program must be listed on this card. The FAC Card is illustrated in Figure 3.

3. Table Length Card (TAL)

This card specifies increments to the minimum data table lengths which were established by the assembler or compiler. The card makes it possible to change at load time the amount of core or drum working area available to the program for a given run. Its inclusion in the Job Request is optional. The card contains a JOB REQUEST ID field identical to those described above, a CARD TYPE field with the

symbol "TAL" to identify the card, and a number of TABLE LENGTH INCREMENT fields. These latter fields are described in Section IV on Facility Assignment. The TAL card is illustrated in Figure 4.

4. Transfer Card (TRN)

This card is included in the Job Request for a program which uses facilities which are to be transferred to the environment of a following program or which are to be received from other programs. The following program requires a TRN card with a RECEIVING COUNT field to accept the facilities. (Section XIII describes some limitations.)⁴ The card contains a JOB REQUEST ID field, a CARD TYPE field with the symbol "TRN", a RECEIVING COUNT field, and a number of TRANSFER fields.

The RECEIVING COUNT field contains one or two decimal digits denoting the number of facilities to be received by this program from one or more previous jobs in this sequence of jobs.

Each TRANSFER field consists of four subfields separated by "slash" (/) characters. These subfields are:

SUCCESSOR ID: the JOB REQUEST ID of the program to which the facility transfer is directed.

If this subfield is preceded by an *, then EXEC itself is the recipient and the remainder of the subfield specifies one of two possible types of transfer:

REQ Facility contains one or more Job Requests

PLB Facility contains a Program Library

Hence, Job Requests or a Program Library may be transferred to the Executive System.

FACILITY TYPE: the type of facility involved in the transfer as denoted by

⁴Facilities that are being transferred to this program are not reflected in the FAC Card requirements.

- a) the symbol "D" for drum storage.
- b) the symbol "T" for a magnetic tape to be transferred after rewinding.
- c) the symbol "N" for a magnetic tape to be transferred without rewinding, or for the transfer of all facility types excluding core, drum, and jump switches.

If the "T" or "N" is preceded by an asterisk, the facility will be given a reserved assignment upon release by the job program or by one of the successive job programs to which it may have been transferred.

FACILITY NAME 1: the symbol by which this job program refers to the facility to be transferred.

FACILITY NAME 2: the name by which the program to which the facility is being transferred, i.e., the receiving program, refers to the facility. If the receiving program is EXEC and the facility contains a Program Library, then this subfield contains the library identity.

The subfields are written in the order

Facility Name 1/Facility Type/Successor ID/ Facility Name 2

on the TRN card. The card is illustrated in Figure 5.

If more than one TRN Card is necessary, they must be grouped together. The order of grouping may be random and the RECEIVING COUNT field may be on any card of the group.

5. Parameter Card (PMn)

A job program may require a set of input parameters to determine or select options of execution. These parameters are entered via the PMn Card. The card contains a JOB REQUEST ID field, a CARD TYPE field with the symbol "PMn" where n is a decimal digit from 0 to 9, and a PARAMETERS field of up to 66 characters. This card is illustrated in Figure 6 and discussed in detail in Section V.

JOB REQUEST ID	CARD TYPE	TABLE LENGTH INCREMENTS
-------------------	--------------	-------------------------

965004	,	TAL	,	MATRIX/2500 , COUNT/100 .
--------	---	-----	---	---------------------------

The allocations for Table Length Tags MATRIX and COUNT are increased by 2500 and 100 respectively.

FIGURE 4: TAL CARD

JOB REQUEST ID	CARD TYPE	RECEIVING COUNT	TRANSFERS
-------------------	--------------	--------------------	-----------

965004	,	TRN	,	4	,	PAY/T/965005/PAYROL , PAYFIL/D/965006/STATCS ,
--------	---	-----	---	---	---	--

	ROE/N/974002/A23 .
--	--------------------

Four facilities are to be received by this program from previous jobs in the sequence. The magnetic tape PAY is to be transferred after rewinding to job number 965005 and is to be referred to therein as PAYROL. (PAY might represent the updated master payroll file which is to become the input to the next payroll maintenance run.)

FIGURE 5: TRN CARD

JOB REQUEST ID	CARD TYPE	PARAMETERS
965004	PM1	(required parameters - up to 66 characters)

The parameters are transferred to the \$PARAM table of the job program. Since EXEC does not edit these parameters, "blank" characters are also transferred.

FIGURE 6: PMn CARD

- Note 1: Information placed in the \$PARAM table at assembly time is not superseded by the parameter card information.
- Note 2: x characters are stored into the \$PARAM table, where $x = 66 (N+1)$ and N is the highest value of n. Eleven words of Fielddata spaces (05) are stored in the table for omitted PMn cards.

C. General Schedule Information

1. Entering Requests

Job requests may be entered into the system at any time. Each job request identity in the schedule at any one time must be unique. Any job request with a nonunique identity will be rejected at time of entry into the schedule. The only exception to the rule occurs when a job request in the schedule is in a suspended state. Here entry of a new job request with the same identity results in the new job request replacing the old one.

The last job request of a sequence of requests must be followed by a job request termination card. The contents of the TERM card are:

- a) Right parenthesis in columns 1 through 8 and 73 through 80.
- b) Plus sign in columns 10 through 22.
- c) All other columns are blank.

2. Indication of Schedule Completion

The completion of the schedule function is signalled by the message:

*jrid*ΔSCHΔACK or INT*jrid*ΔSCHΔACK

The job request ID displayed in the message is that of the last good job request processed. When the job request tables are filled, all job request cards following the last good card are passed over until the termination card is encountered.

3. Job request card errors will result in error typeouts on the console printer. Any error detected will cause the job request in question to be bypassed. The error message format is either:

SPΔ*jrid*ΔERnn or INTΔSPΔ*jrid*ΔERnn

where

SP indicates schedule processor

INT indicates request schedule came to EXEC via an internal transfer

jrid is job request ID

nn is error code

The error codes and their description are listed below:

- | | |
|-----|--|
| ER1 | Job request ID of current job too long. Message shows previous job request ID. |
| ER2 | Card type field too long. |
| ER3 | Illegal card type. |

- ER4 Cards out of order - PM
- ER5 TAL name too long or illegal character on the TAL card
- ER6 TAL number too long
- ER7 Illegal character on TRN
- ER8 TRN field too long
- ER9 FAC name too long
- ER10 Reserve on more than one card
- ER11 Illegal facility requested
- ER12 Illegal character on FAC.
- ER13 Illegal sequence on FAC
- ER14 FAC asked for too large a core block
- ER15 Two FAC have *
- ER16 Job program name too long.
- ER17 Illegal character or sequence on PTY
- ER18 Illegal number of characters in some field on PTY
- ER19 Rerun but no absolute units on PTY
- ER20 Card count on PTY and cards in job do not match
- ER21 PTY cards not grouped together
- ER22 FAC cards not grouped together
- ER23 TAL cards not grouped together
- ER24 TRN cards not grouped together
- ER25 First card of new jrid is not a PTY card
- ER26 Duplicate request I.D.
- ER27 No IBANK specified on facility card
- ER28 Illegal RUSH field on the PTY card or FAC card is not the second card of the job request.
- ER29 Job request core storage table filled.
- ER30 PM number is not numeric
- ER31 Matched priority and precedence of a job in schedule but no sequence number on new job.
- ER32 Matched priority and precedence of a nonsequence job.
- ER33 Priority and precedence of Z63 but not a permanent job.
- ER34 FAC card ended with colon but next card is not FAC card starting with a tag field.

D. Schedule by Paper Tape

In addition to EXEC schedule input for cards, the following paper-tape format is accepted. Characters may be in Fielddata or Flex code.

1. General

- a) The paper-tape header may be any length up to three feet. If something other than master spaces (blank tape) has not been read after three feet of tape, an error message will be produced.
- b) The trailer may be any length greater than one inch.
- c) Each equivalent of a "card" is terminated by a period (.).
- d) The first "card" on tape must start with a carriage return followed by the word SCHEDULE. Any number of characters up to 70 may follow this word. The first "card" is terminated by a period.
- e) The formats for the PTY, FAC, TAL and TRN are the same as defined for cards. The format for the PM card includes another field to define the number of characters which follow. The format is as follows.

jrid, PMn, x, (parameter characters)

if x = a period (.) the parameter characters will be read until a period is encountered

if x = special character (Fieldata code 76, Flex code upper case S), the 66 characters after the comma will be taken as parameter information

if the x field is missing, reading will stop.

- f) The stop code (Fieldata code 57, Flex Code 43) is used to define end-of-data, except in the parameter characters on the PMn "card".
- g) Ten consecutive master spaces (one inch of blank tape) after any legal character and before the stop code will cause termination of reading with an error message.
- h) A backspace code (Fieldata 77, Flex Code 61) may be used at any time to cancel the characters typed so far for a particular "card". (Except in the parameter characters of the PMn card.)

2. Fieldata Code

All Fieldata codes are legal on the PMn "card". The following characters are legal for the PTY, FAC, TAL, and TRN "cards":

05 through 37 (A-Z). 60 through 71 (0-9).
41 (-), 50 (*), 53 (:), 56 (.), 74 (/), 75 (.), and 44 (=).

All other characters will be ignored on these "cards".

3. Flex Code

All the rules that apply to Fielddata code apply to Flex code.

The following are the characters allowed and their Fielddata equivalents (note that upper and lower case punches are necessary in some cases). Unless noted otherwise, all characters are lower case.

a. PTY, FAC, TAL, and TRN cards

Alphabetic - lower case letters a - z
 Numerics - lower case numbers 0 - 9

<u>Flex character</u>	<u>Flex code</u>	<u>Fielddata equivalent</u>
Upper case - (dash)	56	-
+	54	*
- (dash)	56	:
.	46	.
(vert. slash)	5Ø	/
.	42	.
=	44	=
back space	61	back space
stop code	43	stop code
tape feed	ØØ	master space
space	Ø4	space

b. PM Card

All characters as shown above plus:

<u>Flex character</u>	<u>Flex code</u>	<u>Fielddata equivalent</u>
upper case S	24	special
upper case U	34	upper case
upper case L	11	lower case
tab	51	tab
carriage return	45	carriage return
upper case)	42)
upper case P	15	+
upper case H	Ø5	<

<u>Flex character</u>	<u>Flex code</u>	<u>Fielddata equivalent</u>
upper case G	13	>
upper case N	Ø6	<u> </u> (underline)
upper case D	22	\$
upper case (46	(
upper case T	Ø1	"
upper case Q	35	?
upper case E	2Ø	!
upper case A	3Ø	! (apostrophe)
upper case C	16	;

Code delete 77 is ignored on any card type.

E. Deletion of Job Requests

It may be necessary to delete a job from the schedule before or after it has been initiated. To accomplish this, the operator types in a request of the form:

```
DEL jrid or DEL*jrid
```

where DEL identifies it as a request to delete a job request from schedule

jrid represents the alphanumeric JOB REQUEST ID which was associated with this job on all of the job request cards.

* is optional and causes the remaining jobs in the sequence, if any, to be deleted.

The EXEC response to this message will be one of the following:

DEL ACK	- job has been deleted
DEL RUN	- the job request cannot be deleted because it is operating (the operator must use the "TER" message to terminate the running program.)
DEL Ø6 IN ERROR	- job request ID not in schedule.

III. SELECTION

There are two phases to the Selection function of the Executive System. The first phase involves selecting, according to priority and precedence relationships, a candidate for initiation from the job schedule. The second phase is an allocation check to determine if the facilities required by the candidate are available. If they are available, the job program is loaded and initiated. If they are not available, then another candidate is selected. This process is continued until all jobs in the current priority class have been checked for possible initiation. When this occurs, the Selection function terminates its operation.

A. Operation

The Selection function operates whenever one of the following changes occur:

1. Job Requests have been added to the schedule
2. A job program, or EXEC itself, has released a facility from its operating environment.
3. The operator has made a facility available.
4. A job program has terminated and all of its facilities have been released and made available for reassignment.
5. A job program has just been initiated.

Following selection of a job, control is given to the Loading function of EXEC to load and initiate the program. When the job has been initiated, control reverts back to Selection which attempts to select another job.

B. Method of Selection

Seven factors are used to select the next candidate from the job schedule. They are: RUSH designator; priority assignment, in the order A to Z from highest to lowest; precedence within a priority, in the order from 0 to 63; sequence assignment, in the order 1 to 63; Run Type; and Mix Type.

All of this information is made available in the Job Request PTY Card as described in Section II.

1. Permanent

If the permanent indicator is set as a result of a P in the RUSH field of the PTY card, the job is skipped regardless of other information.

The permanent job may be selected by the use of the second version of the SEL unsolicited message (see table 11). Thus resulting in assignment of a running jrid, priority, precedence, and sequence to the permanent job. The original permanent job remains in the schedule and can be removed only by use of a DEL unsolicited message.

2. RUSH

When a RUSH job is detected, all currently operating jobs using the needed core facilities are interrupted and the RUSH job is loaded and initiated.

3. Priority

All jobs receive a priority from A to Z, A being the highest. If priorities are omitted from the job request, EXEC assumes a "Z" priority. All jobs, or sequences of jobs, in given priority class must be initiated before a job from the next lower class will be considered as a candidate. This scheme of stepping within priority class may be altered by operator intervention. A type-in of the form:

TPR

will cause the Executive to "terminate priority restriction." The priorities will then be examined sequentially until a candidate is selected or until the end of the job schedule is reached. To return to the scheme of initiating all jobs within a given priority class before examining the next class, the operator may "initiate priority restriction" by means of the type-in

IPR

4. Precedence

Precedence assignment ranges from 0 to 63. If precedence is not assigned in the Job Request, EXEC will assume a value of 63. The jobs in a priority class will be examined sequentially according to precedence value until a candidate for initiation is selected. When the precedence values are equal, candidates are examined in the order in which the job requests were submitted.

5. Sequence

The sequence assignment serves to specify the order in which a set of associated jobs are to be run. The jobs are associated by assigning the same priority and precedence values to each of them. This association can be used for a set of jobs where the output of one job is used as input for the next job, or for a set of jobs which are to be tested.

Sequence numbers range from 1 to 63. All jobs with identical priorities and precedence values are run serially with respect to each other in the order specified by their sequence numbers. The sequence numbers do not, however, have to be consecutive. If the sequence numbers of two or more associated jobs are identical, these jobs will be run concurrently if facilities permit.

Following completion of a job (or jobs, if two or more ran concurrently) in sequence, the following job in sequence is initiated if facilities permit. If the following job in the sequence does not pass the facility availability check this fact is remembered and the job is always the first candidate for selection following any release of facilities. After the first job program in a sequence has been initiated, the following job requests are given top priority regardless of the current priority level.

This concept of scheduling permits the definition of more than one set of jobs to be run in sequence at the same priority level. The facilities assigned to one job may be transferred to a succeeding job in the sequence. The ROC program output tape from SLEUTH or from an 1107 compiler may be used as the input tape to EXEC for testing the assembled programs. (See Section XIII).

Although jobs in sequence are run serially with respect to each other, they still may be defined for parallel, serial, or external rerun types of operation, with respect to all jobs not in this sequence. There is, however, one restriction; external reruns may only be initiated at the beginning of a sequence and are always initiated serially.

When a job program in sequence is terminated, with or without error prior to its normal completion, the operator has the option of deleting the remaining jobs in the sequence. (See Section XII.)

6. Serial Selection

For a given priority, jobs to be run in serial will be selected as candidates for initiation when no job programs are currently operating, or after all jobs to be run concurrently have been initiated. At this time the operator is notified to load the program tape on an assigned tape unit if it is not already loaded. The serial job will not be loaded, however, until all currently operating programs have terminated.

7. Mix Type

All job programs that are I/O limited in their operation will be initiated into a mix of concurrently operating programs before more than one computer-limited job program is initiated. This check applies to all programs run under EXEC control except for succeeding job programs in a sequence of jobs.

C. Facility Check

Once a candidate for initiation has been selected, a check is made of the Facility Availability Table to determine if the facilities required by this program are available. (The required facilities are submitted in the Job Request FAC card.)

In order to determine the availability of the requisite facilities, the following questions are asked:

1. Are there enough channels available for each type of facility?
2. Are there enough units available for each channel?
3. Are there enough consecutive locations on each requested drum channel?
4. Does the core bank with the largest group of consecutive 2048-word blocks have enough to satisfy the larger of the two core requirements (IBANK or DBANK).⁵
5. Is there a block of locations in the other core memory bank large enough to satisfy the smaller of the two requirements? If not, is there enough left in the assigned block of the first core bank checked.⁵

If the answer to any of these question is no and at least one job program is currently running, the job request will be temporarily bypassed as a candidate for initiation. If the answer to any of these questions is no, and no job programs are currently running, the job request will be suspended and the timeout:

jrid FAIL FAC SUS.

will occur:

⁵When a 65K (or 49K with 32K in the first bank and 16K in the second) core memory is available and a check 4 fails, a check is made to determine if a block of core extending into both core banks exist which is large enough to satisfy both IBANK and DBANK requirements. If the first part of check 5 fails the existence of such an overlapping block of core is checked before the final check in 5 is performed. See Section V. on Loading of job programs for discussion of IBANK and DBANK.

D. Selection Interruption

In order to provide the computer operator with some measure of control over Executive System operation, EXEC will accept the type-in:

HSL

which causes it to halt the process of selecting jobs for initiation. In order to cancel this order and to direct EXEC to resume the selection of jobs for initiation, the type-in

SEL

is given. This type-in has meaning only when it follows an "HSL" message.

A more exacting degree of control is allowed by specifying a job request ID with the HSL type-in. This causes the specified job request to be suspended as a candidate for selection until the computer operator reactivates the job request with the type-in

SEL jrid (See Table 10)

Job requests suspended by EXEC after passing the facility check will not be eligible as candidates for selection until the operator reactivates them with the SEL jrid message.

E. EXEC Message Relating to Job Selection

When all unsuspended jobs in the schedule have been selected, the following message is typed:

SCH SELECTED

When the schedule is selected and no jobs are running, the following message is typed:

SCH COMPLETE

When a job which meets the requirements of the facility check has been selected but there is no available UNISERVO from which to load the job program, the following message is typed out:

*jrid*NO AVAIL INPUT TAPE

IV. FACILITY ASSIGNMENT

Object programs require a certain environment in which to operate. This environment is some subset of the total facilities comprising the 1107 system. Since one or more programs may be operating and using some facilities when a new program is being initiated, all facilities will have to be referred to by the programs in a symbolic manner. The Executive System will assign an available facility to every symbolic facility referenced by the program and listed as a required facility on the FAC card. Available facilities are those not used by EXEC or by any other program.

The various facilities to be assigned by EXEC may be grouped into three major classes:

1. Core memory which includes:
 - a. an instruction storage block
 - b. a data storage block
2. Drum memory for each program segment storage and data storage.
3. I/O equipment which includes:
 - a) magnetic tape
 - b) paper tape reader
 - c) paper tape punch
 - d) card reader
 - e) card punch
 - f) High-Speed Printer
 - g) console selective jump switches

These facilities are uniquely assigned to a job program for its use. They are returned to an available status when the job program terminates its operation or when it releases a facility (see IV. E. below).

The 1107 Assembly System produces two outputs which are significant for facility assignment by the Executive. One of these, the facility description data which appears on the listing output of the assembly of the job program, is used to prepare the FAC card of the job request. The Executive System checks the information on the FAC card to determine if the required facilities are available. If they are, the second of the two outputs, the modification record of the assembled program, is used to assign the required facilities. Limited changes in the facilities required in the job program may be made via the job request. The modification record is part of the ROC program output of the assembly.

The FAC Job Request cards are discussed below. The Modification Record is described in Section V.

A. Facility Descriptions (FAC Card)

The form of the FAC card was illustrated in Figure 3.

The card is identified by the symbol "FAC" in the second field. The first field contains the alphanumeric identity of the job request.

The remaining columns of the card are divided into fields separated by commas. These fields contain the facility requirements organized by unit tape, channel, and number of units. The last field may be followed by a period, comma, or blanks. Additional FAC cards with the same format may be included. If all unit tapes of facilities to be deleted will not fit on the current FAC card, the card can be terminated by a colon in which case the remaining tag subfields must be included as the first entries on the next FAC card.

The FACILITY REQUIREMENT fields take the general form

```
*ff cc/uuuuuuu : s:s,
```

where

* identifies the logical channel with which the tape containing the job program is to be associated.

ff is a 2 letter code defining the type of facility (see below).

cc are two decimal digits in the range 0-15 denoting the logical channel number. When cc=0 or is omitted, it implies that no specific channel associations are necessary for the units specified in the corresponding units field. This channel designation should be used only when the number of units specified is greater than one.⁶ When cc≠0, logical channel numbers 1 to 15 serve to associate those units which will operate efficiently with each other on a common channel.⁷

uuuuuuu is a one to seven decimal digit subfield specifying the number of peripheral units, registers, or the number of 2048 word core-memory blocks required for the corresponding facility.

⁶This concept does not apply to drum channel requirements.

⁷Note that these channel assignments are logical and do not necessarily correspond to physical 1107 I/O channels.

s is the symbol for a facility that is normally required by the job program, but which is not to be assigned for this run. This unit is not included in the count of units required on this logical channel. This subfield is not applicable to core or drum facilities.

The FAC cards describe the required facilities in terms of numbers and logical channels. The same information together with the symbol for each facility is contained in the Modification Record of the job program. The two descriptions must match. The symbol "MT 1/2" on the Job Request FAC card conditions EXEC to expect a minimum of two symbolic tape references to be defined in the Modification Record for logical channel 1.

The asterisk (*) identifies the logical tape channel with which the magnetic tape containing the job program should be associated for efficient operation. This specification will be honored if the job program tape is not already loaded. If it is already loaded, then an attempt is made to allocate the logical channel requirement to the channel containing the job program tape. The count of the number of units required for a channel marked with an asterisk must include the tape containing the job program.

The possible values of the ff, cc, and uuuuuuu sub-fields are indicated in Table 1.

TABLE 1: FAC CARD FIELDS

TYPE OF FACILITY	ff	cc	uuuuuuu
UNISERVO IIIA	MT	0 to 15	1 thru 16 units
UNISERVO IIIC	MC	0 to 15	1 thru 16 units
UNISERVO IIA	MA	0 to 15	1 thru 12 units
Card Reader	CR	0 to 15	1 unit
Card Punch	CP	0 to 15	1 unit
High-Speed Printer	HP	0 to 15	1 or 2 units
Paper Tape Reader	PR	0 to 15	1 unit
Paper Tape Punch	PP	0 to 15	1 unit
Magnetic Drum	MD	0 to 15	Number of required drum locations.
Data Bank	DB		Number of 2048 word blocks of core memory required for data storage.
Instruction Bank	IB		Number of 2048 word blocks of core memory required for instruction storage.
Selective Jump Switches	JP		1 thru 15

NOTE: All values of uuuuuuu are given for one required channel. Additional Card Readers, etc., may be assigned to other logical channels.

The repeated FACILITY REQUIREMENT fields must be organized according to the following rules:

- 1) Fields must be grouped according to facility type.
- 2) The fields of a given type must be listed according to the amount of units or memory locations required, with the largest requirement listed first.
- 3) The logical channel zero requirement, if present, must be listed as the last requirement for that facility type.

The efficiency of the facility availability check performed by EXEC will be increased if the facility requirements most apt to be unavailable are listed first on the job request FAC cards. This is left to the discretion of the user.

B. Changes in Facility Requirements

1. Peripheral Equipment (excluding drum)

Certain types of programs (the 1107 SLEUTH Assembly System, the 1107 SORT/MERGE Program) have facility requirements which vary according to the type of operation and other factors. To accommodate programs of this type, EXEC will recognize certain variations in the requirements definition fields of the FAC Card. These variations may specify modifications which are to be made to the program's facility requirements as was established at assembly time. The modifications to be made fall into three categories: partial deletion, complete deletion, and minimum requirements.

- a. **Partial Deletion:** This is accomplished by identifying those units (tape, printer, card reader, etc.) which are not to be assigned to this job for this particular operation. As an example, consider the requirement defined by the field

MT 2/5

This states that logical tape channel 2, as defined in the Modification Record of the job program, must contain exactly 5 units assigned to the program. If two units are not needed for this run of the program, then the field can be changed to read

MT 2/3 : LIST : NOLIST,

where "LIST" and "NOLIST" are the symbolic unit tags for the units which are not to be assigned. Console selective jump switches can be deleted only in this manner. The comments in the following two paragraphs (b. and c.) do not apply to jump switches.

- b. **Complete Deletion:** This is accomplished by omitting the complete field from the FAC Card. For example, if all tapes on logical channel 2 in the above example are not to be assigned to this job program for this run, then the field MT 2/5 would not be listed on the FAC card. This is legal only if none of the five tapes on logical channel 2 are listed in the Modification Record of the program as part of the minimum equipment configuration.
- c. **Minimum Requirement:** Each unit symbol listed in the Modification Record of the job program may be defined in the source code as part of the minimum operating requirements of the program.

Suppose that 3 of the tape units assigned to logical channel 2 in the above example were defined as belonging to the minimum configuration. Then the field "MT 2/5" could be written as "MT 2/3" on the FAC Card if only the minimum configuration for logical channel 2 is required.

If, after the deletions have been made, a logical channel requires more units than were requested in the job request, the loader attempts to make the additional assignments. This does not apply when complete deletion was requested by omitting the field from the FAC card. If the additional facilities can be assigned, the operator is notified that some type of I/O equipment was updated and loading is continued. When the required number of units are not available, loading is terminated with a message on the typewriter indicating insufficient I/O available. The job request is suspended, if part of a sequence, or deleted by the EXEC and a new job may be initiated.

2. Core and Drum (TAL Card)

The fields of the FAC card which define the number of drum and core table locations may be changed as required within the limits set by the program. The minimum number of required locations for each drum channel and for core table storage is calculated by SLEUTH. This number may be increased but never decreased. This number is used to determine in advance of program loading if there are enough drum and core locations available for use by the program. If on loading it is found that the program demands more drum registers than the number specified in the job request, an attempt will be made to allocate that amount. If the attempt is successful, the operator will be notified that the drum requirement in the job request has been updated and loading will be continued; if the additional amount is not available, the job request will be suspended, if part of a sequence, or deleted with an appropriate message on the typewriter and a new job will be considered for initiation. The drum requirements for a particular logical channel need not appear on the FAC card if all drum tables on the channel are assigned via external transfers to the program.

If any of these drum or core requirements are to be changed on loading, there must be one or more table length (TAL) cards (see Section II) present in the job request. This card permits EXEC to receive and transmit to the EXEC Loader a new length definition for a limited number of tables in the job program. This information is determined as assembly time from information in the job program's source code.

The TAL card is illustrated in Figure 4. The TABLE LENGTH INCREMENT fields are of the form

1/iiiiiii

where 1 is a core or drum Table Length Tag as defined in the source code and consists of from 1 to 6 alphanumeric characters; iiiiii is a 7 digit decimal number which is the increment to the length of the table.

When the TAL Card is used to increase the length of selected core and drum tables, the overall core and drum requirements will be changed. This change must be reflected in the FAC Card.

C. Allocation of Facilities

Facilities will be allocated to job programs according to the following rules. The word "channels", as used in these rules, is limited to those channels which contain the same type of peripheral equipment. Note that card reader and card punch are different types.

All channel assignments are made on the basis of the facility requirements indicated on the FAC cards. Unit assignments within the assigned channel are made at load time on the basis of the information contained in the Modification Record of the program.

1. Peripheral Facilities (excluding drum)

- a. Channel Assignment: The channel assignment procedure takes into consideration the MIX TYPE of the program. Channel assignments for I/O limited programs are taken entirely, if possible, from channels that currently do not contain units assigned to I/O limited programs. If this is not possible, or if the program is compute-limited, then this check is not made and the units are assigned normally.

The logical channel with the largest unit requirement will be assigned first to the channel with the largest number of available units. Then the logical channel with the second largest unit requirement will be assigned to the channel that, prior to the first assignment, had the second largest number of units available. This process is continued until all logical channels have been assigned to physical channels.

Units not associated by channel will be assigned to available channels in a manner designed to leave, if possible, an equal number of available units on each channel.

- b. Unit Assignment: The symbolic unit definitions contained in the Modification Record of the job program are assigned in order of definition to available units, in ascending sequence, on the previously assigned channel.

When assigning tape units, those that have been defined at assembly as input units are assigned to the lowest numbered available units that are currently rewound with interlock.

2. Magnetic Drum Facilities

- a. Channel Assignment: Drum register requirements are treated as one consecutive block. The size of this block is considered in the same manner as are the number of units per channel described above in paragraph 1a.
- b. Drum Address Assignment: The block of registers (defined in the Modification Record) required by the program is assigned to the smallest available block of drum registers large enough to hold the register requirement. All drum DTABLE assignments are made from this area at load time.

3. Core Memory Facilities

- a. Core Bank Assignment: The single core requirement, or the larger one if there is more than one, is assigned to the core bank with the largest number of available 2048 word blocks. If two core requirements are present, the other one is assigned to the alternate core bank or to the same bank if the alternate bank does not have the required availability.⁸
- b. Core Assignment Within a Bank: Core memory locations are assigned to job programs in consecutive increments of 2048 words. This allows efficient utilization of the memory lockout capability. The core requirement is assigned to the smallest available group of 2048-word blocks that is large enough to satisfy the requirement. These blocks are assigned to the program in ascending sequence if they are contained in core bank 1 and in descending sequence if in bank 2. This procedure promotes an available area of core which extends into both core banks.

⁸ For an 1107 with a 65K (or 49K with 32K in bank I and 16K in bank II) core memory, the Executive program itself is stored in the lower addresses of core bank 1 and its data tables are in the upper addresses of core bank 2. This arrangement leaves the balance of core memory as one continuous block available for assignment to job programs. The procedure for core-bank assignment is the same with these exceptions: if there is an available block which extends into both banks and the single core requirement cannot fit in either bank (or if the two core requirements cannot be assigned to alternate banks), then an attempt is made to assign the total core requirements into the block of available registers extending into both banks before assigning the total requirements to one bank.

D. Operator Notification

After all facilities have been assigned, the Loader will type out those assignments which are necessary to the operator during loading and execution of the program. If the facility requirements of the program being loaded cannot be met by the Loader, a message is typed indicating those types of facilities which could not be assigned. Section V contains a complete description of the typeouts concerning facility assignments as well as other messages from the Loader.

When facilities are not all assigned, either because of an insufficient statement or because of the lack of additional available facilities to meet the actual requirement, EXEC either deletes the job request if not part of a sequence or, if part of a sequence of jobs, EXEC temporarily suspends the corresponding job request from the list of possible candidates until the operator either: 1) deletes the request or 2) reactivates the request with the SELjrid message. (See Section III.D.) The assigned facilities are restored to an "available" status and an attempt is made to select another job for initiation.

E. Release of Facilities

When the equipment used by a program is no longer required, it should be released by the program so that it may be reassigned to other programs. To release a unit of peripheral equipment, the address of a Transfer Packet specifying the unit to be released is loaded into a designated film memory register and control is transferred to EXEC via a Load Modifier and Jump (LMJP) instruction. EXEC will return the unit to "available" status. Once a unit has been released by a program, it cannot be regained except under special circumstances as described in Section XIII.

The facilities which may be released through this function of EXEC are:

- Paper Tape Reader
- Paper Tape Punch
- Card Reader
- Card Punch
- Magnetic Tape Unit
- High-Speed Printer
- drum assignment

The calling sequence for release of a facility is:

8	9	FUNCTION	14	15	SUB FIELDS
		L D B			\$ Q Ø , t r , , \$ U O P
		L M J P			\$ B 1 , \$ R E L
		J U M P			, C

tr is the address of a transfer packet defining the facility to be released (see Section XIII). C is the address of a contingency routine in the job program.

\$REL is the entrance to EXEC for release or transfer of facilities. EXEC returns control to the second line following this call after executing the desired release. In the unusual case where the facility is being transferred to EXEC or another program, and the transfer table is full, the REL packet cannot be accepted. Control is then returned to the first line following the call.

If a program requires a unit of peripheral equipment, such as a paper tape or card reader, for reading in initial data, it should release this unit when its function has been accomplished. This may permit other programs requiring such units to be initiated at an earlier time.

When a program is to be terminated, all facilities assigned to the program will be automatically released by EXEC, with the exception of those which are to be transferred to a succeeding program (see Section XIII). Hence, the procedure described above is used only when releasing facilities during intermediate stages of a program operation.

F. EXEC rules for return of facilities to an available status are:

- 1) Core - no special rule, core areas are made available
- 2) Drum - listed in priority order:
 - a) if the "external transfer" indicator is set, the facility is assigned a transfer status
 - b) if "down" indicator is set, the facility is put in a down status
 - c) the facility is made available
- 3) Remaining peripherals
 - a) if "transfer" indicator is set, the unit is placed in a transfer status and is not available for assignment to programs
 - b) if "down" indicator is set, the unit is placed in a down status
 - c) if unit is a UNISERVO that contains a library tape temporarily assigned to program, the unit is reassigned to a library assignment

- d) if "reserved" indicator has been set, the unit is placed in a reserved status and the operator is notified with the message:

Ccc Δ UuuΔRESERVED

- e) if none of the above is true, the unit is released to the available pool
- f) when a UNISERVO is placed in a reserved or available status, it is rewound without interlock if it is extended; if the program was abnormally terminated, the UNISERVO will be rewound with interlock, if not already rewound with interlock

Programs should normally handle their output unit by rewinding them with interlock and notifying the operator with a "change" or "unload" function.

A. Location of Job Program

Before a job program can be loaded, its location must be specified and communicated to the Executive System. This is done via information contained in the PROGRAM MEDIUM and MEDIUM NAME fields of the Job Request PTY Card. The different types of information that can be contained in these fields are indicated in Section II. Table 2 describes various entries which may be contained in these fields and the actions taken by EXEC in each case.

EXEC selects unassigned tape units from the highest numbered unloaded units on the channel that currently has the greatest number of available units. Consequently, when the job program is loaded, the unit it is being loaded from, may also be assigned to one of the job program's symbolic tape requirements. For this reason EXEC will rewind the job program tape with interlock and, if it has been assigned to the job program, the operator will have the opportunity to load it for the job program. This is not true the case where the tape symbol is equated to a symbolic program reference (see Table 2 for t=p). Here the tape unit is not rewound.

If the tape referred to by the symbol t is also to be assigned to the job program (equated with the p symbol). EXEC assigns the t tape to the highest numbered available unit on the channel to which the p tape has already been assigned. If the t tape is a currently assigned library tape, then the unit requirement for the asterisked logical channel requirement is reduced by one and an attempt is made to assign the unit to the physical channel containing the t tape. If this fails, a normal allocation is attempted. After the job program has been located, EXEC will read its ID block to verify its existence.

In order to direct the operator to place a named tape on a selected unit, EXEC uses a typeout of the form,

LOAD△Ccc△uu△WITH△t

where t is the symbolic tape name, and is bracketed with two box (□) symbols if t is defined as a program library tape
 cc is the decimal channel number from 0 to 15
 uu is the decimal unit number from 0 to 15

when the operator has conformed to this order, he must type in

Y

If the operator cannot conform, he types in

N

and EXEC will suspend the job request and select another job. (See Section III.D.)

The UNISERVO IIIC tape unit instruction repertoire does not include a backward read. Therefore the backward read search is replaced with a rewind and forward read search. The tape position is maintained as with IIA and IIIA. However, when it is lost or not supplied, the UNISERVO IIIC will be positioned at load point following loading of the ROC program.

Two problems may be encountered when EXEC is locating the job program on the assigned input tape. They are:

- 1) The job program cannot be located
- 2) Nonrecoverable trouble on input tape.

TABLE 2: LOCATION OF JOB PROGRAM (EXECUTIVE)

PROGRAM MEDIUM	MEDIUM NAME	EXECUTIVE ACTION
*	CccΔUuu	EXEC positions the tape (after rewinding) on the assigned unit to the specified job program. A tape containing a rerun dump must be specified in this manner.
T, C, or A	t	EXEC selects an available tape unit and instructs the operator to load the tape identified by the t symbol. After the job program has been transferred to core by the Loader. EXEC rewinds the tape with interlock. The tape unit containing the program tape is still available for assignment to the job program being loaded
T, C, or A	t = p	The procedure for this case is identical to that described above with one exception. In the above case the tape unit assigned temporarily to t is still available for assignment to the job program. It may be assigned to any of the symbolic tapes referenced in that program. In this case, however, the tape defined by the symbol p in the job program receives the same absolute assignment given to the t symbol. This feature is useful when data is to be stored on the job program tape following the job program.
T, C, or A	*t	The asterisk identifies the t tape as a library tape containing a file of programs. EXEC checks the Program Library Registration Table to determine if this program library tape or drum file is currently assigned. If it is assigned, the program location information of the PTY card is redundant. If the library is on drum, the drum library directory table is searched to obtain the exact address of the job program (see Section XIII.F.4). If the library is on tape, the tape is positioned to the ROC program. If it is not already assigned. EXEC selects a tape unit from that channel which currently contains the largest number of available units and directs the operator to place the t tape on the selected unit. This tape and its assignment is then entered into the program library registration table.

PROGRAM MEDIUM	MEDIUM NAME	EXECUTIVE ACTION
T, C, or A	*t/rr	The two-digit field rr denotes the position of the job program on the t tape. This position is compared with the current position of the tape to determine the direction in which to search for the job program. When the job program is found the new relative position of the tape is recorded in the Program Library Registration Table.
T, C, or A	*t=p	During the time that the program library is assigned to the job program, it is not available to EXEC for the purpose of loading job programs.
T, C, or A	*t=p/rr	This is the most general case and is a combination of the above procedures.

When EXEC cannot locate the job program, EXEC rewinds the input tape without interlock and the operator is interrogated with the message:

tape name C_ _ΔU_ _ΔNO Program name.

Legal replies are:

- 1) "Y" EXEC searches the tape again
- 2) "N" job request for program in question is suspended in the schedule
- 3) "Other" the message is repeated for nonsense replies

When a nonrecoverable trouble occurs, EXEC attempts to rewind the input tape without interlock and the operator is interrogated with the message:

tape name CccUuuΔI/OΔER*ΔΔ*

The legal replies are:

- "Y" operator has corrected the condition and EXEC is to try again
- "N" EXEC suspends job in schedule
- "Other" the message is repeated for nonsense replies

The above "N" replies are acknowledged with the message

*jrid*ΔSUSΔER2

When EXEC suspends a job program, the typeout:

*jrid*ΔSUSΔERx

occurs on the console printer. The value of "x" indicates the condition which caused suspension. These conditions are as follows:

x interpretation

- 0 The absolute channel and/or unit defined as the one containing the ROC program to be loaded is not in this configuration
- 1 The absolute channel and unit denotes a facility which is not a UNISERVO tape unit.
- 2 The operator has replied no, "N" to one of the messages concerning the load or positioning of the tape supposed to contain the ROC program.
- 3 The facility transfer storage table contains more transfers to program than were allowed for in the RC field of the job request "TRN" card

- x interpretation (cont.)
- 4 The job program to be loaded is not listed in the Drum Library Directory (see Section XIII. F. 4.)
- 5 Job program is listed in Drum Library but label block cannot be found.
- 6 Nonrecoverable drum error when searching Drum Library Directory
- 7 Nonrecoverable drum error reading directory entry or the label block of the ROC program
- 8 Absolute channel and unit defined as containing input ROC program is not available or reserved.

B. Location of Subroutines for Complex Programs

The subroutines to be incorporated into a complex program at load time are located in a manner similar to locating the main job program. Location is specified on the PTY card in the LIBRARY MEDIUM and LIBRARY NAME fields. These fields are further described in Table 3.

TABLE 3. LOCATION OF SUBROUTINE LIBRARY

Library Medium	Medium Name	Executive Action
*	Ccc Uuu	This is the absolute channel and unit for the subroutine library. The unit must be available or reserved.
T, C, A, or Z	t	If "t" is currently registered as a tape program library and is mounted on a UNISERVO, that assignment is given to the EXEC Loader. If "t" is not registered as a program library, an available unloaded UNISERVO is selected by EXEC and the operator is instructed by the message: LOAD CccΔUuu WITH t to load the subroutine library tape. In the latter case, the UNISERVO containing the subroutine library is still assignable to the job program being loaded. It may not, however, be used as the segment storage tape. If the library medium field is a Z, then the main program tape is a library tape and contains both the main program and the associated subroutines. In this case the medium name field is ignored.

The location of subroutines must be in accordance with the following rules:

- 1) The library of subroutines must be stored on magnetic tape.

- 2) The subroutines must not be on the same library tape as the main job program (in this case the main program cannot demand segment storage).
- 3) The subroutine library tape title can not be equated to a symbol in the job program.
- 4) Subroutines are loaded in the order of occurrence in the library hence the tape position is not maintained.
- 5) If the subroutine library tape has not been assigned by load time, the EXEC will assign it to an available unit for the duration of the loading process. This same physical unit is available for assignment to the job program being loaded.

C. Operation and Main Functions of Loader

Job programs scheduled for operation under control of the Executive System will be accepted for loading in relocatable (EXEC ROC) form only. ROC programs to be loaded by the EXEC Loader are classified as either simple or complex programs. Either type may be segmented. A simple program is one which is entirely contained on one input medium and in one program file. All subroutines referenced by a simple program are incorporated at assembly time. A complex program is composed of a main program and one or more subroutines. The main program is contained in one program file while each subroutine is contained in an additional program file.

Programs which are to be run under control of the 1107 Executive System and which utilize the Executive I/O Functional routines are initiated via a job request submitted to the Executive System. The Executive System will read the first block on the input medium specified by the PTY card of the job request; this block is the program identification record. The Executive System will then transfer the peripheral equipment location of the input medium, the number of words in the block just read, and the core address of the block to the Loader. If subroutines are to be incorporated at load time, their input medium is also passed to the Loader.

The Relative Load routine utilizes the Executive I/O Functional routines in order to perform its own input/output operations. The words generated by an assembler for a program segment are modified and read into core at the execution position. For each IBANK or DBANK area of the object program, one block is written on tape or on drum if storage is specified. When a program segment is written on a storage medium, that area of core to which it is assigned is cleared. Any section not requiring storage remains in core and the initial operating section; i.e., the first program segment to be executed must, therefore, be loaded last or not be written over by a succeeding section. After initial modification, loading, and storage of the program by the Relative Load routine, the program itself must read succeeding segments.

The main function of the loader are as follows:

- 1) Accept relocation data and use it to modify and load the object program.
- 2) Modify core data table lengths and magnetic drum table lengths to fit definitions introduced at load time.
- 3) Change all symbolic I/O, drum, and jump switch references to absolute assignments.
- 4) Assign the absolute addresses for all symbolic references to the Executive System.
- 5) Assign all core data tables and load input parameters into the data area of the object program.
- 6) Store program segments as directed by the program being loaded.
- 7) Incorporate subroutines into the program at load time.

Figure 7 illustrates the operation of the EXEC Loader.

1. Format of Program File

An EXEC ROC program file is produced by the 1107 Assembly System, or by the appropriate compiler, and consists of the object program in the form required by the Relative Load routine together with all of the necessary information for purposes of loading and modification. For a more detailed description of an EXEC ROC program file, the CLAMP manual (UP 2575) should be referenced.

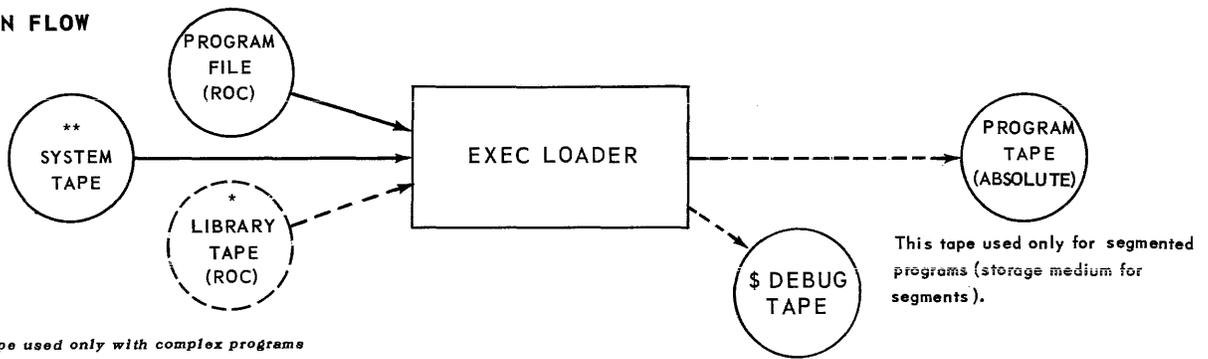
Each ROC program file is composed of four types of records. These records appear in the following order:

- 1) Identification record
- 2) Modification record
- 3) Program record
- 4) Termination record

Each record consists of one or more blocks as required by its specific function.

All but three types of tags are eliminated from programs which are assembled in Relative Object Code. These tags are:

INFORMATION FLOW

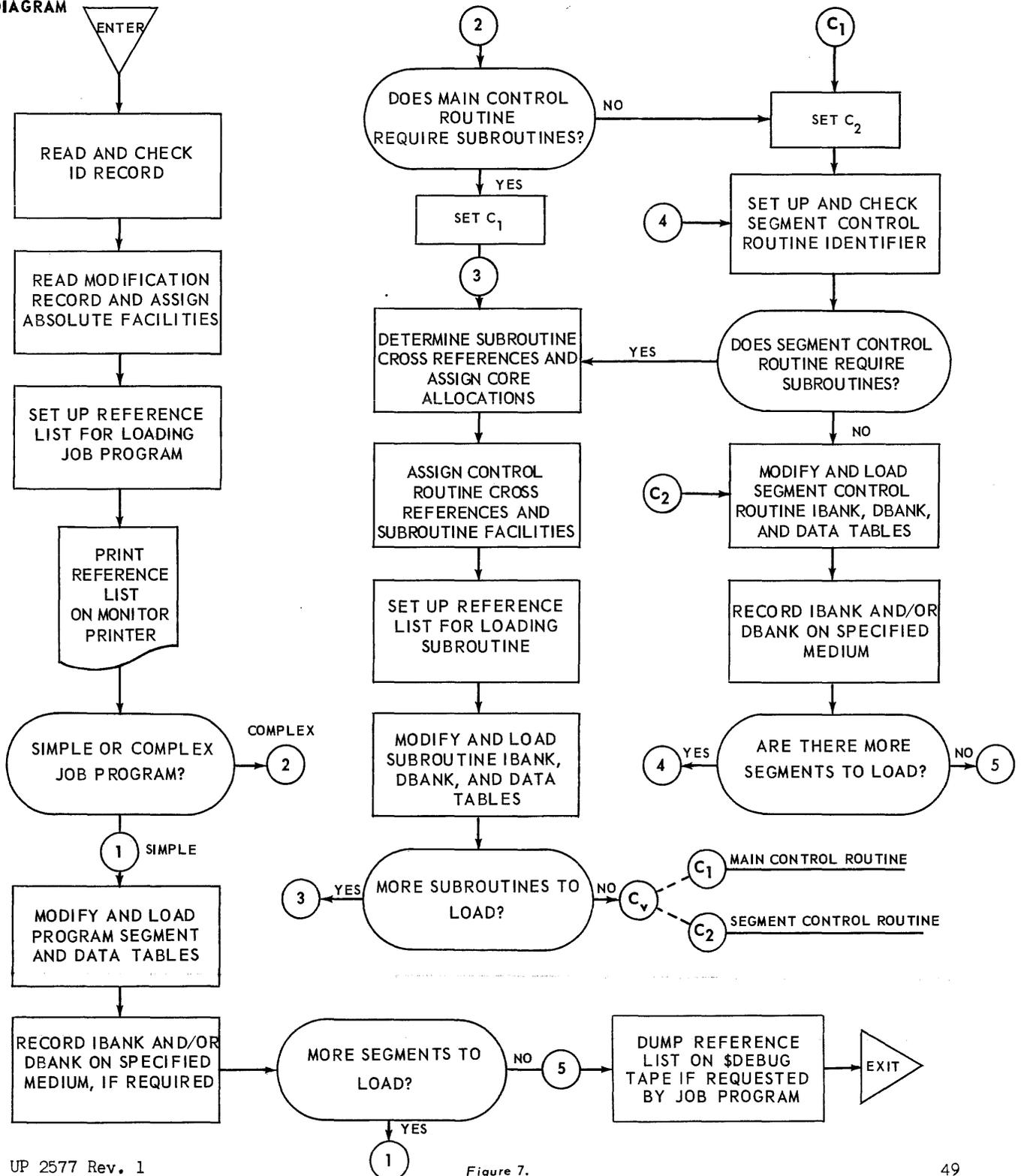


N * This tape used only with complex programs

** Same tape as Executive System tape

NOTE: MAGNETIC DRUM MAY BE SUBSTITUTED FOR ANY (OR ALL) TAPES EXCEPT LIBRARY TAPE AND \$DEBUG TAPE

BLOCK DIAGRAM



- 1) Program Name: This is the unique symbolic name (12 Fielddata characters) assigned to the program instruction portion of an object program.⁹

The program name is used by the Executive System and/or the Relative Load routine to identify the program and to serve as the relative zero address of the instruction portion (IBANK) of the program. The absolute assignment for this tag is given to the Relative Load routine by the Executive System. This absolute assignment then becomes the first location assigned to the program instructions in core memory. All tags other than the program name which might appear in the IBANK of the source program are given relative assignments with respect to the program name.

- 2) Data Table Tags: Each ROC program file may contain one or more data table tags. This tag serves as the relative zero address of a data table. The absolute assignment for this tag becomes the first location assigned to the data table in the core memory or the magnetic drum memory. All other locations within the table are made relative to the data table tag.
- 3) Data Table Length Tag: Each data table tag in the ROC program file may have a data table length tag associated with it which represents the minimum length of the data table. The table length may be increased at load time if a length tag was defined at assembly time. The absolute assignment given to this tag becomes the current length of the data table or core or drum memory.

The four types of records which make up an object program file are briefly described in the remainder of this section.

- a. Identification Record

The identification record contains the program name and a code identifying the class of program. The program is classed as either a simple program or a complex program; either class can be segmented or nonsegmented. A subroutine may be any program which is not segmented.

⁹ For object programs produced by SLEUTH, this is the symbolic label which is obtained from the 6-character tag of the PRO line of the source program. This tag is expanded to 12 characters with the right-most 6 characters filled with space codes.

b. Modification Record

This record contains four types of tables which contain information necessary to modify the object program for its operating environment. The types of tables that may appear in the modification record are:

- 1) input/output references
- 2) system references
- 3) drum references
- 4) core references

As the source program is assembled, each symbolic reference within the program is assigned a reference number. These reference numbers replace all symbolic references in the program record.

The tables of the modification record contain lists of the symbolic references of the source program together with the reference numbers which have replaced them.

The tables of the modification record are used by the Relative Load routine to generate a reference list which will include the absolute assignments for all symbolic references. The reference list is arranged in four sections corresponding to the four tables of the modification record. The reference numbers of the modification record correspond with entries in the reference list. The operation of the Relative Load routine with respect to the modification record is discussed in Section IV.C.2. The following paragraphs contain a brief description of the contents of the modification record.

- (1) **Input/Output References:** This portion of the modification record contains a list of all symbolic references to the peripheral equipment.¹⁰ The references are 2-word entries. The first word contains the symbolic reference for the peripheral equipment. For I/O equipment, the second word contains an equipment type field denoting the type of unit (UNISERVO IIIA tape unit, card reader, etc.), and a logical channel grouping. In the case of jump switches, the word contains a logical switch number.

¹⁰ With the exception of magnetic drums.

- (2) System References: Four separate tables may be included in this section of the modification record. These tables include Symbolic references to the Executive System, undefined symbols used in subroutines, and lists of associated subroutines for the object program.
- (a) Executive System References: A list of 1-word symbolic references to the Executive System, the Executive I/O Functional routines, and the Relative Load routine.
 - (b) Subroutine List: A list of the program names of all subroutines referenced by the program but not incorporated into it at assembly time.

These symbols are used at load time to load the required subroutines from the library tape. This list is contained in complex programs and subroutines only.
 - (c) External Reference List: This portion of the system references table lists symbolic entrances to the subroutines in the subroutine list. The absolute address for these entrances are assigned as the subroutines are loaded from information contained in the library tape directory. This list is contained in complex programs and subroutines only.
 - (d) Entrance List: This table contains a list of the symbolic entrances to the program other than the program name. Each 2-word entry contains the mnemonic symbol for the entrance and the address of the entrance relative to the subroutine program name.
- (3) Drum References: This table contains a list of all data table tags and data table length tags which reference the magnetic drum(s) together with the assigned minimum length for each drum table.
- (4) Core References: Each 3-word entry in this table contains a core data table tag, its associated data table length tag, and the assigned minimum length of the table.

c. Program Record

All object program instructions and data words are contained in the program record. This portion of the object program file is arranged in two parts: segment sections and data table sections.

- (1) Segment Sections: A segment section contains the set of instructions that comprise an object program segment as well as the associated data words (DBANK), if any. From the point of view of the source program, the program segment contains all words generated in the IBANK and DBANK.

¹¹ In SLEUTH the subroutine names are indicated in the XREF line of the source program.

- (2) Data Table Sections: These parts of the program record contain the set of words for each variable-length data table (DTABLE) assigned at assembly time. The EXEC Loader will place these data table words in the core memory locations allocated to the table.

Both the segment sections and the data table sections contain two types of words: program words and modification indicator words. The program words are those instruction or data words which are generated by the Assembly System from the information contained in the source program. These words contain fields such as function codes which are fixed by the source program, and fields such as core memory address references which are to be modified by the Relative Load routine. The modifiable fields will contain reference numbers which refer the Relative Load routine to the reference list which is generated from the information contained in the modification record.

The modification indicator words contain fields to denote the type of modification necessary.

- d. Termination Record

This record is the last block of the program record. It contains the address of the first object program word to be executed, relative to the program name. (This is the address given in the ENDPRO declarative of a SLEUTH source program.)

2. Program Modification Techniques

- a. Modifiable Fields

For programs assembled in Relative Object Code, certain fields within the generated program words are necessarily incomplete and must be modified at load time. From the source program, an assembler must construct modification indicators describing how a field is to be modified. The assembler must also produce the modification record which defines the source-code tags from which the modifiable fields are generated.

The Relative Load routine modifies references in program words to reflect internal program references, core data table references and their associated table lengths, drum table references and their associated lengths, I/O references, and external references, i.e., those pertaining to subroutines and other programs.

- (1) Internal Program References: Modifiable fields which contain internal program references are converted to their absolute form by adding the current address assigned to the Program Name. The word forms and word fields that can be modified are indicated in Table 4.

TABLE 4. MODIFIABLE FIELDS

WORD FORM	MODIFIABLE FIELD(S)
	BIT POSITIONS
Whole Word	15-0
Half Word	15-0, 33-18
Instruction Word	15-0
I/O Access Word	15-0, 33-18
Variable Format Word	15-0, 33-18

- (2) Core Table References: The modifiable fields of the ROC program which contain core table references are of the forms indicated in Table 5. With the exception of the third and fourth forms shown, the reference numbers for the current tag assignments are contained within the field being modified. In the fifth form, the current value of the field is the sum or difference of the values associated with the reference numbers contained in that field.

TABLE 5. MODIFIABLE FIELD FORMS

1	Data Table Tag
2	Data Table Length Tag
3	Data Table Tag \pm Constant
4	Data Table Length Tag \pm Constant
5	Data Table Tag \pm Data Table Length Tag

In the case of the third and fourth forms shown, the current value of the field being modified is the sum or difference of the value associated with the reference number and the constant. In all field forms of this type, the constant is contained in the field being modified. The tag reference number is either in the field being modified or in the indicator field associated with that program word, depending on the value of the constant.

For core table references, both the tag reference number and the constant are in the field being modified if the constant is less than 2^6 . If the constant is equal to or greater than 2^6 , the tag reference number is carried in a 9-bit area of the associated indicator field.

- (3) **Drum Table References:** Magnetic drum references are actually input/output references. They are discussed separately, however, because the manner in which drum tables are used parallels that for core tables.

The modifiable field forms for drum references are as shown in Table 5. The constant and tag reference number are in the field being modified if the constant is less than 2^{12} . For other values of the constant the tag reference number is in a 9-bit area of the indicator field associated with the program word.

The drum table length tags are handled differently depending on their position in the program word. Length tags which occur in the right half of the program word are replaced by their current assignments in a 23-bit field (positions 22-0) at load time. When a length tag occurs in the left half of a program word, it is replaced by the 16 least-significant bits of its current assignment (in positions 33-18 of the program word) at load time. The most significant seven bits of the assignment are ignored.

Drum references are discussed again in the following paragraphs.

- (4) **Input/Output References:** The absolute assignments for I/O references are made by the Executive System for EXEC ROC programs.
- (a) **Peripheral Equipment:** A single symbolic reference is used to denote both the unit and the channel for the peripheral equipment. At load time, this symbol is replaced by a 30-bit field (positions 29-0) which contains the current channel and unit assignment. The channel assignment is in positions 29-26.
- (b) **Drum:** For drum data table tag references, the current drum address is consigned to bit positions 22-0. Both of these assignments are shown in Figure 8. Note that bits 23-16 of the program word for peripheral equipment references are not modified. The contents of this field can therefore be assigned at assembly time. The unit address is indicated by master bit selection.

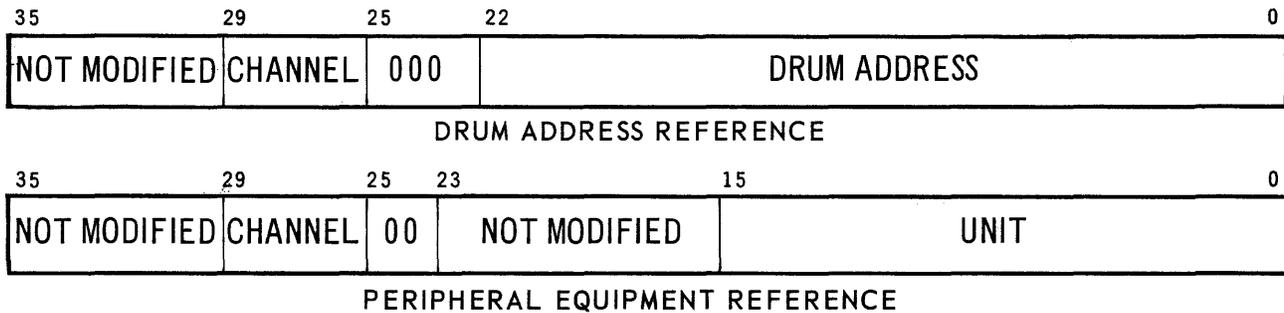


Figure 8. EXEC ROC References

Two or more I/O references may be equated at assembly time.

- (5) **Selective Jump Switch References:** The symbolic selective jump switch references are replaced by a 4-bit absolute switch number in positions 25-22 at load time. Each jump switch used must be given a unique symbol. Up to 15 different selective jump switches may be specified.
- (6) **External References:** Object programs which operate under control of the Executive System will make reference to it or to the Executive I/O Functional routines during their operation. These references will be in the form of specific system tags (e.g., \$XIO) and will include I/O requests, closeout operations, and error procedures.

At assembly time, the system tags are replaced by reference numbers to a system symbol table in the reference list and the actual mnemonic tag is stored in that table. The reference numbers are replaced with absolute assignments at load time.

b. **Data Table Conventions**

The core area assigned to an object program is divided into three sections: the instruction section (IBANK), the fixed-length data section (DBANK), and the variable-length data section (DTABLE). The fixed-length data section contains items such as constant pools and fixed-length tables. Like the instruction section, the fixed-length data section is one continuous block of core whose length is determined at assembly time.

Data tables in the variable-length data section of core or drum are defined by a unique symbolic data table tag and by a data table length tag. The length tags need not be unique for each table but only one value must be assigned for each unique length tag. The value assigned to the length tag at assembly time is the minimum length of the table. This minimum value may be incremented at load time by specifying a table length increment in the job request.

The Relative Object Code defines each data table as an independent table. Symbolic internal table tags are not allowed. Provision is made for equating the starting addresses of two or more tables. If this is done, the subject tables are listed consecutively in the reference list produced by the Relative Load routine (see below). The minimum length tag must be specified at assembly time.

The variable-length data section of core may contain as many tables as is desired subject to the conventions discussed in the preceding paragraph.

c. Reference List

The Relative Load routine generates a reference list which contains the absolute assignments for all symbolic references. The reference list is composed for four sections corresponding to the sections of the modification record.

Each reference list section contains a maximum of 128 one-word entries except for the drum reference section. The latter contains 2-word entries (for a maximum of 256 words). The four sections of the reference list are:

- 1) I/O references
- 2) system references
- 3) drum references
- 4) core references

d. \$PARAM and \$ERROR Tables

Two special tables are defined for ROC programs. These are the input parameter table and the error table and are designated by the table names \$PARAM and \$ERROR respectively.¹²

- (1) \$PARAM Table: A program may require a set of input parameters to determine or select options of execution. Accordingly, the job request may contain one or more input parameter records. These records (PMn cards, Section II) contain 11 words of 6 alphanumeric characters each. A maximum of 10 such records (cards) are permitted in an object program. They must be identified sequentially as "PMn" where n is a decimal digit from 0 to 9. N+1 records are stored in the \$PARAM table, where N is the highest value of n. Omitted records numbered below N are taken as blanks. The PMn card is illustrated in Figure 6, Section II.

¹²

The table names "\$PARAM" and "\$ERROR" must be present if these tables are to be acceptable to the EXEC Loader.

The input parameter words are transferred into the \$PARAM table of the object program if, and only if, such a table was defined at assembly time. This table must be defined in the source program by using the reserved label \$PARAM as a data table tag. If the \$PARAM table exists, it is assigned as the first data table in the program data bank. The parameter words are placed in the \$PARAM table prior to loading the program words. This means that program words defined in the \$PARAM table at assembly time will replace information read from the parameter cards into the same locations. A reserved area in the table causes only those locations not filled with parameter words to be set to zero.

If the \$PARAM table is not large enough to accommodate the input parameters, or is not present, and an attempt is made to load such parameters, then an appropriate typeout will notify the operator that the load has been terminated.

- (2) \$ERROR Table: Each program to be run under Executive System control contains locations in the \$ERROR table corresponding to the error interrupts (core-memory addresses 192-199). These locations contain entrance addresses to error recovery subroutines within the object program. The \$ERROR table also contains carry and overflow indicators and an image of the film memory registers.

Entries for the \$ERROR table may be specified at assembly time. All words with no specified entries are cleared to zero except those listed below.

Location	H ₁	H ₂
\$ERROR+9	IBANK Length	IBANK Starting Addr.
\$ERROR+10	DBANK Length	DBANK Starting Addr.

The \$ERROR table is assigned storage in the instruction area (IBANK) of core. The \$ERROR table is a fixed-length table (97 locations) and is always assigned in the last 97 locations of a program IBANK. If the program requests an \$ERROR table length greater than 97, the load is terminated with the proper error indication. This table need not be defined in the job program unless it is referenced by the program. (See Appendix K for \$ERROR table layout.)

3. Facility Assignment Notification

All facilities required by the main program, except drum tables used for segment storage, are assigned prior to loading the program words of the main program or loading subroutines. When the facilities have been assigned, the operator is notified through the facility assignment notification typeout.

The message contains the following information:

- 1) Core memory assignments
- 2) Selective jump switch assignments
- 3) I/O equipment assignments
- 4) Warning information indicating those types of facilities which were updated (more facilities assigned than requested in the job request).

The message format is as shown below:

jrid program-name Ixx/yy Dxx/yy *DRM

*JSW jj/symbol jj/symbol etc.

jj/symbol jj/symbol etc.

*Ccc uu/symbol uu/symbol etc.

*Ccc uu/symbol uu/symbol etc.

When facilities are updated by the Loader, the operator is warned via additional information in the typeout. This information is underlined in the format shown above and will not normally appear. The meanings of the symbols used in the Facility Assignment Notification are as follows:

jrid is the 6-character alphanumeric identity of the job request.

program-name is the alphanumeric tag assigned to the main program file being loaded.

xx is the 2K block number where the I(IBANK) or D(DBANK) starts in memory. The beginning block numbers for banks 1 and 2 are 00 and 16, respectively.

yy is the length, number of 2K blocks, assigned to the IBANK or DBANK.

*DRM is the warning message that indicates drum facilities were updated (more drum registers required than requested in the job request).

symbol is the symbolic reference used in the program to reference jump switches or peripheral units

JSW is the symbolic notation for selective jump switch assignments. An asterisk preceding this symbol indicated that switches were updated (more switches required than requested in the job request).

jj is the current jump switch assignment. This is a decimal number from 01 to 15.

- C is the symbolic notation for I/O channel assignments. An asterisk precedes this symbol when some type of peripheral equipment has been updated.
- cc is a decimal number from 00 to 15 denoting the current I/O channel assignment.
- uu is a decimal number from 00 to 15 indicating the current peripheral unit assignment.

The selective jump switch assignments are printed four to a line.

Each different channel in the I/O equipment assignments starts a new line. The unit assignments associated with a specific channel are printed four to a line.

When insufficient facilities are requested in the job request and the Loader is unable to assign the additional facilities, loading is terminated with the message.

jrid program-name INSUF (code)

where (code) indicates those types of facilities which were insufficient.

The (code) field will contain one or more of the following symbols:

- 1) JSW - insufficient jump switches
- 2) I/O - insufficient peripheral equipment
- 3) DRM - insufficient drum registers
- 4) DB - insufficient DBANK

When the load is terminated for any reason, the job request is deleted or suspended by the EXEC. If the job is part of a sequence, it and the sequence are suspended.

If additional peripheral units must be assigned to a subroutine during the loading of a complex program, the operator will be informed by a message on the typewriter. The format of the message is

jrid SUBR I/O Ccc uu/symbol

where cc is the absolute channel assignment, uu is the absolute unit assignment, and symbol is the symbolic reference used in the subroutine to reference the I/O unit.

The operator is requested to mount magnetic tapes in two cases during the load. One is when the program demands segment storage on tape and the other is when the program requests that the reference list be dumped on the \$DEBUG tape (see MIDAS, UP 3846). Both requests are communication LOAD functions and must be answered by the operator.

When a segment storage tape is to be mounted, the message has the form:

LOAD Ccc Uuu

where the channel and unit have been assigned to a symbolic I/O reference shown on the Facility Assignment Notification typeout. This request can only appear once and applies to the first tape storage segment encountered during the load. The Loader attempts to write the segment regardless as to whether the message reply is "Y" or "N".

When the Loader stores program segments on magnetic tape, the tape is left as positioned after writing the last segment. If the storage occurred on a UNISERVO IIIC tape unit, it is the responsibility of the job program to write an end-of-file mark after the last block to insure that a run-away tape situation does not occur. This happens when an attempt is made to read past the last block of information and no end-of-file tape mark is present.

When the \$DEBUG tape is to be mounted, the following message is typed:

LOAD Ccc Uuu WITH \$DEBUG

If the message reply is "N", the reference list is not written on the \$DEBUG tape.

4. Error or Warning Notification

Error or warning conditions other than those connected with insufficient facility requests may be detected during the load. The loading may be stopped prior to the Facility Assignment Notification typeout if certain errors are encountered. When an error or warning condition occurs, the operator is informed by a message on the console printer of the form

LD *jrid* ERR nnn or LD *jrid* WAR nnn

where nnn is a 3 digit error or warning code. The detection of warning (WAR) conditions is never cause for terminating the load. The detection of one error (ERR) condition always causes the load to be terminated, however, up to five error or warning messages may be typed before the load is terminated.

At least one of the five messages will be an error message. The error or warning codes are given below.

- ØØ error in format of ROC program file
- 01 first block of program file not label block
- 02 checksum error on reading program file
- 03 attempted to assign a segment storage drum table via a transfer or to transfer the assignment of a segment storage drum table to a succeeding program
- 04 not EXEC ROC program
- 05 table length modification not allowed
- 06 jump switch cannot be deleted
- 07 I/O facility cannot be deleted
- 10 illegal systems reference in main program
- 11 nonrecoverable I/O condition
- 12 machine or peripheral equipment error
- 13 program file does not contain symbol or drum table being transferred to program
- 14 program file does not contain symbol of drum table requested for transfer to a succeeding program
- 15 program file does not contain symbol of I/O facility being transferred to program
- 16 program file does not contain symbol of I/O facility requested for transfer to a succeeding program
- 17 I/O facility requested for transfer to a succeeding program was deleted
- 20 table (DALL) containing drum table assignments full, cannot transfer drum table to succeeding program.
- 21 requested DBANK area too small
- 22 requested IBANK area too small
- 23 \$PARAM table too small
- 24 segment storage tape assigned to same unit as main or subroutine program file
- 25 no DBANK requested for load of a complex program
- 26 subroutine requested segment storage
- 27 tape from which subroutines are to be loaded is not a library tape
- 30 library tape format error
- 31 library directory too large for 2K buffer supplied by Loader
- 32 subroutine or external reference not in library directory
- 33 total drum area requested not enough to include segment storage.
- 34 attempted to transfer more than one I/O facility or drum table to same symbolic reference in program
- 35 external reference or subroutine referenced by a subroutine not found in subroutine library

- 36 subroutine medium not defined
- 37 subroutine jump switch not defined in main program
- 40 unable to assign subroutine I/O facility
- 41 illegal systems reference in subroutine
- 42 subroutine drum or core table not defined in main program
- 43 subroutine drum or core table longer than length of corresponding table on main program
- 44 subroutine IBANK or DBANK longer than length given in INFO block of library
- 45 warning, unable to make all table length changes specified on TAL card (program does not contain one or more of the table length tags)
- 46 segment storage medium deleted
- 47 attempted to transfer to an independent drum table, a drum area less than the area specified in a dependent table (a dependent table is one which has the same starting address as the independent)
- 50 core table length increment on TAL card too large (16-bit sum is less than original length)
- 51 requested \$ERROR table length greater than length (97) fixed by EXEC.

5. Peripheral Equipment and Drum Table Assignments

I/O equipment and drum tables are either independent or dependent. A dependent facility is one which has been equated to an independent facility. A dependent drum table has the same starting address as the independent table to which it is equated. I/O equipment which is optional may be deleted at load time. Facilities may be assigned via external transfers.

The rules which govern facility transfers, I/O equipment deletions, and drum table allocation are:

- 1) I/O equipment transfers to program
 - a) If the transfer is for a dependent facility, only that facility is assigned.
 - b) If the transfer is for an independent facility, the assignment is given to that facility and all its dependent facilities except those which received prior assignments via transfers.
 - c) The transfer of an assignment to a facility previously assigned via a transfer is an error.

- 2) I/O equipment deletions
 - a) Only those facilities defined as optional may be deleted.
 - b) If the deletion is for a facility already assigned via a transfer, the deletion is ignored.
 - c) If the deletion is for a dependent facility, only that facility is deleted.
 - d) If the deletion is for an independent facility, it and all dependent facilities which were not transferred are deleted.
 - e) If the deletion is for a facility which has been previously deleted, the deletion is ignored.
 - f) If the deletion is for a facility which cannot be found, the deletion is ignored.
- 3) I/O equipment transfers from program - The transfer of a facility which was deleted is an error.
- 4) Drum table transfers to program
 - a) If the transfer is for a dependent table, only that table is given the allocation.
 - b) If the transfer is for an independent table, the allocation is given to it and all dependent tables except those which previously received transferred allocations. (Each dependent table must have a length equal to or less than the length of the transferred allocation.)
 - c) The transfer of an allocation to a drum table previously assigned via a transfer is an error.
 - d) The transfer of an allocation to a segment storage drum table is an error. (However, a segment storage table may receive a transferred assignment if it is a dependent table.)
- 5) Segment storage drum tables
 - a) A table dependent upon a segment storage drum table is not given a valid assignment.
 - b) An independent segment storage table is assigned when the segment is loaded. The table length is equal to the length of the segment.
 - c) A segment storage table which is dependent upon a table not used for segment storage is assigned prior to loading the segment. The table length must be large enough to allow storage of the segment.

- 6) Program references to drum tables
 - a) Independent tables not used for segment storage and those tables dependent upon them may be referenced at any point in the coding.
 - b) Tables dependent upon segment storage tables do not receive valid assignments and should not be referenced.
 - c) Independent segment storage tables should not be referenced within the segment using the tables and in succeeding coding. (Subroutines within the segment using the table should not reference the table.)
- 7) Drum table transfers from program

The transfer of a segment storage drum table assignment to a succeeding program is an error.

6. Loading and Modification of Complex Programs

Provision is made in the EXEC Loader for the incorporation of subroutines into an object program at load time.¹³ A program requiring the addition of subroutines is called a main program. The main program and its associated subroutines is called a complex program. Complex programs may, or may not, be segmented.

a. Subroutines

The Relative Object Code for subroutines must have certain characteristics in order to facilitate their incorporation into a program at load time.

- (1) Directory Card: A directory card is produced for each subroutine when it is assembled. This record contains the following information in a compact list which becomes a part of the library tape directory and is made available to the Relative Load routine:
 - (a) Subroutine name¹⁴
 - (b) Names of all subsidiary subroutines referenced by this subroutine.
 - (c) Length of IBANK for this subroutine
 - (d) Length of DBANK for this subroutine
 - (e) Subroutine entrances
 - Symbolic entrance (tag)
 - Entrance address relative to the subroutine name.

¹³ The Retrieval section of the 1107 LIBRARIAN is an integral part of the Loader.

¹⁴ The subroutine name is analogous to the program name of the main program.

- (2) **Library Tape:** The library tape necessary for loading subroutines can be generated in two ways:
- (a) By submitting the information of above list and the EXEC ROC subroutine file of each subroutine to the 1107 LIBRARIAN. See 1107 LIBRARIAN Guide, UP 2579 Rev.1, for the procedure necessary to generate the library tape.
 - (b) By submitting the EXEC ROC subroutines to the 1107 ELF routine (Element Filing routine). No lists of information as shown above need be supplied. ELF builds an EXEC ROC library tape using only the subroutines as input. The procedure necessary for generating the tape in this manner is found in the 1107 ELF Programmer's Reference Manual.
- (3) **Symbol Definition:** Each symbol used in a subroutine must be defined by that subroutine. The subroutine must consist of an IBANK area, and may have a DBANK area and/or one set of variable length tables. The core data tables (DTABLES) of the main program, other than the \$ERROR table, are loaded into the first locations of the area received from the EXEC for the program DBANK.

- (a) **DBANK:** When a subroutine is to be in core at all times (not stored in a segment), the DBANK is assigned to the locations preceding the fixed-length data section (DBANK) of the referencing master program. The object program is thus given a single fixed-length data section.
- (b) **Common Data Tables:** All data tables used by the subroutine must be defined in the subroutine's variable-length core data section. All drum tables used by the subroutine must be similarly defined within the subroutine. All tables thus defined in either the variable-length core data section or on the magnetic drums are considered as common data tables.

The referencing main program must also define every common data table which is defined in a subroutine. All common core data tables must be defined in the variable-length data section of the main program. Common drum tables must be defined in the drum table section of the main program.

- (c) **I/O Equipment:** Like data tables, all I/O equipment used in common by a main program and the subroutines referenced by it must be defined in both the main program and the subroutine. I/O equipment which is used only by a subroutine need only be defined within that subroutine. The facility requirements for the total object program however, must include all I/O equipment referenced by the subroutines. This is done by requesting enough extra units, of the proper type, on logical channel zero to satisfy the subroutine requirement for that type.

- (d) Subroutine Entrances: Provision is made in the Relative Object Code for multiple entrances to the subroutine. All of these entrances, excluding the subroutine name, must be defined symbolically as well as relative to the subroutine name. The Relative Object Code for subroutines contains a special section for these entrances.
 - (e) External References: The main program and the subroutines it references may contain symbolic entrances to other subroutines. A location within a subroutine may be referenced by an external reference plus or minus a constant.
- (3) Restrictions: Subroutines may not be segmented. All segmentation occurs within the main program.

b. Nonsegmented Complex Programs

The modification record of the main program is loaded first and the reference list is generated. The main program subroutine list (generated from the XREF line of the source code) is then submitted to the retrieval section of the 1107 LIBRARIAN. The subroutines are then loaded from the library tape followed by the program record of the main program.

- (1) DBANK and Data Tables: Each complex program has only one fixed-length core data section. The fixed-length data section DBANK of the main program is preceded by the fixed-length data sections DBANK of all referenced subroutines. All common core data tables, i.e., tables in common by both main program and subroutines, are placed in the variable-length core data section of the object program. All of these common core data tables must be defined within the main program. These definitions are incorporated into the subroutines during loading. The minimum table lengths specified as assembly time are honored. If the table length as specified in the main program is less than that given in the subroutine, an error indicator is presented.

The data table tags defined in the main program are perpetuated throughout the entire loading procedure. All common data tables in the referenced subroutines, are therefore defined in terms of the main program specifications. The data table tags defined within the subroutine are used only during loading and are not perpetuated. If a data table which was not defined in the main program is encountered during subroutine loading, an error indication will be given.

The \$PARAM and \$ERROR tables, when present in the main program, are treated as common data tables. All drum data tables are also considered as common tables and are handled in the same manner as are core tables.

- (2) **Instruction Section:** The instruction section (IBANK) of the main program is preceded by the instruction sections of the subroutines. The object program will then contain only one instruction section after loading. All references to the main program are specified relative to the main program name.
- (3) **I/O References:** All common I/O references, i.e., those used in common by both main program and subroutines (or by two or more subroutines), are defined in the main program. These definitions are incorporated into the subroutines as they are loaded. Logical channel assignments specified in the main program are followed. Logical channel assignments in the subroutines are therefore not honored.

Common I/O references are specified by using the same symbol in the main program and in the subroutine or by equating symbols in the main program. The I/O references defined in the main program are perpetuated throughout the loading procedure. This is not the case with I/O definitions within subroutines, however, and these are used only during subroutine loading.

The I/O equipment necessary for the operation of a complex program is specified in the main program I/O facility requirements. These facilities include common I/O equipment as well as equipment used by a subroutine alone.

c. Segmented Complex Programs

Provision is made for loading segmented complex programs. The subroutines within the program are never segmented; only the main program may be segmented. In segmented complex programs, the main program is divided into a main control routine and one or more segment control routines.

- (1) **Main Control Routine:** This section of the object program must remain in core at all times during execution of the segmented complex program. This routine provides the control necessary for transferring program segments into core for execution. Subroutines are included in the main control routine in the same manner as for nonsegmented complex programs discussed above.

The main control routine contains all definitions associated with the entire main program. These definitions include:

- (a) All common core and/or drum tables used by the main control routine, the segment control routines, and/or the associated subroutines.
- (b) All common I/O references used by the main control routine, the segment control routines, and/or the associated subroutines.
- (c) All noncommon I/O references used by the master control routine and/or the segment control routines.
- (d) The subroutine names of all subroutines associated with the main control routine.
- (e) The entrances to subroutines (other than subroutine name) associated with the main control routine.

The subroutines associated with the main control program are loaded and remain in core at all times during execution.

These subroutines may be referenced by the main control routine, the segment control routines, or by any of the subroutines within a segment.

The main control routine may reference any segment control routine as well as any subroutine associated with the main control routine. It may not, however, reference a subroutine associated with a segment control routine.

- (2) **Segment Control Routine:** Each segment of the main program contains a string of instructions which may be augmented by one or more subroutines (incorporated at load time) and which control the subroutines contained within that segment. This group of instructions comprises the segment control routine.

The segment control routine and all associated subroutines must be recorded on a program specified storage medium after they have been converted to absolute form. If the storage medium is drum, it must only be referenced in the segment being stored (excluding subroutines) and succeeding coding.

In the segment control routine, the IBANK of all subroutines to be included are located following the main program portion of the segment control routine. The subroutine IBANKS will be in the order of occurrence of the subroutines on the library tape.

The segment control routine contains definitions for the subroutine names and other entrances for all subroutines associated with it. Each segment control routine which has associated subroutines must specify a fixed-length core data section (DBANK) if storage of the DBANK(s) for the subroutines is desired. This section must include at least one word. The DBANK of the segment control routine is located relative to the DBANK of the master control routine.

The DBANK of all associated subroutines is then located following the DBANK of the segment control routine.

The segment control routine may reference the main control routine or any other segment control routine. It may also reference any subroutine with it or with the main control routine. It may not, however, reference a subroutine associated with another segment control routine.

If the DBANK(s) of subroutines associated with a segment control are relatively small, storage may not be desired. When a segment control has no DBANK, the DBANK(s) of the subroutines are loaded following the last UNSTORED word in the DBANK, and the last word of the subroutine DBANK becomes the last UNSTORED word. "Last UNSTORED word" is the highest-addressed data table or DBANK word remaining in core and not stored on an external medium.

When using this method in loading, it must be kept in mind that over writing may take place since all segment control DBANK(s) are located relative to the first location of master control DBANK.

If a DBANK is defined in some segment controls and not others, allowance must be made for the area used by subroutine DBANK(s) in those segment controls where DBANK is not specified.

The suggested way to make use of this feature is to place all data in master control DBANK and define no DBANK(s) in segment controls.

D. Initiation of Job Program

When a job has been loaded and prepared for operation, the Executive System will type out the following message:

MIX. jrid₁ Δ jrid₂ Δ ⑤ jrid₃

where jrid is the job request ID of the PTY card. The ID's listed correspond to all jobs currently operating or temporarily halted. The presence of a stop symbol "Ⓢ" preceding the job request ID indicates that the program is loaded and waiting for operator initiation by entering the message:

PRO jrid

If the stop symbol is not present, the program is already operating.

The presence of an asterisk (*) preceding either the stop symbol or the job request identifier indicates that the program has been terminated but the EXEC has not yet processed the termination.

VI. INTERRUPT

The interrupt section of EXEC processes all external request, input data termination, output data termination, function termination, real-time clock, and external synchronization interrupts. The re-entry address, i.e., the address of the instruction in the program which was about to be executed when an interrupt occurred is stored in film-memory register \$BØ (address 000000). This register is therefore reserved for the sole use of the Executive System.

EXEC decodes the type and channel number of the interrupt and routes it accordingly. The external request, input data termination, output data termination, and function termination interrupts, except for those on the channel used for typewriter and keyboard, will be used by the input/output section of EXEC to control the execution of all requested I/O functions. The interrupts which occur on the typewriter and keyboard channel will be used by the communication section of EXEC to control the transfer of data and messages to or from those devices.

Each job program to be run under EXEC control contains locations in its \$ERROR table corresponding to the error interrupts (core memory address 192-199). The locations which correspond to interrupts which the job program is prepared to handle must contain entrance addresses to error recovery subroutines within the program. The locations which correspond to interrupts which the job program is not prepared to handle must contain a zero. EXEC will cause an operating program to be terminated with informational memory dump in the case of interrupts which it is not expecting.

The actual core memory error interrupt locations (192-199) will contain references to the Error Interrupt section of EXEC which will determine the location in the program's error image which corresponds to the detected interrupt. The contents of the Program Address Counter at time of interrupt will then be stored at the address specified by the program's error image and control will be transferred to this address plus one. Eight unassigned film memory registers (location 000120 through 000127 octal) are reserved by EXEC for processing the error interrupts.

The computer's real-time clock register and the associated real-time clock interrupt will be reserved for use by the Executive System. When control is relinquished to any job program the real-time clock is set to some

predetermined value to insure that the Executive always regains control after a desired time interval. This prevents programs from retaining control of the computer for an unreasonable length of time as a result of infinite loops or failure to return control as expected. This feature also allows the system to check on operator type-ins. Once the operator has initiated a type-in on the keyboard, characters must be received within a certain time interval following receipt of the preceding character. Thus, in the event the operator fails to terminate a type-in, or to terminate it in the manner expected by the Executive System, the operator can be notified to take the necessary corrective action. Otherwise, in such a situation EXEC could wait indefinitely for the completion of a type-in.

The External Synchronization Interrupt will not be used by the Executive System and may not be used by programs operating under Executive control.

For nonrecoverable error interrupt conditions, one of the following termination codes and the contents of the P register are stored in the BØ film-image register for the terminated program:

- 01 - illegal operation
- 02 - trace mode
- 03 - memory lockout (no recovery)
- 05 - characteristic underflow
- 06 - characteristic overflow
- 07 - divide overflow
- 51 - illegal operation and illegal recovery address in \$ERROR table
- 52 - trace mode and illegal recovery address in \$ERROR table
- 55 - characteristic underflow and illegal recovery address in \$ERROR table
- 56 - characteristic overflow and illegal recovery address in \$ERROR table
- 57 - divide overflow and illegal recovery address in \$ERROR table
- 61 - illegal trace mode set in area other than current program or EXEC entrance

<u>Location</u>	<u>Error Interrupt</u>	<u>Captured P-register</u>
\$ERROR + Ø	Not assigned	Not applicable
\$ERROR + 1	Illegal function code	CI + 1
\$ERROR + 2	Trace Mode	JUMP TO address*
\$ERROR + 3	Memory Lockout	CI + 1
\$ERROR + 4	Not assigned	Not acceptable
\$ERROR + 5	Characteristic underflow	CI + 2
\$ERROR + 6	Characteristic overflow	CI + 2
\$ERROR + 7	Divide overflow	CI + 2

*EXEC subtracts one from the captured address and stores this value in the first instruction of the trace mode subroutine of the job program.

The trace mode can be used concurrently by any other number of the job programs operating under EXEC control. Trace mode should be set dynamically by the job programs utilizing it (by execution of the STJP instruction). When a job program relinquishes control to EXEC, either voluntarily or by occurrence of an interrupt, and trace mode is currently set, EXEC sets an indicator for the job program. Prior to returning control to a job program, the trace indicator for the job program is checked, and if set, trace mode is set on the jump to the job program re-entry point.

The following comments should be considered when using trace mode:

- 1) A jump to an EXEC system reference (\$X10, \$REL, etc.) which results in a trace mode interrupt is handled as any other jump during trace mode, i.e., store the referenced address in the first instruction of the trace mode subroutine and exit to the subroutine. Unless the trace mode subroutine exit is changed by the job program, the system will be honored.
- 2) If it is desirable to set the trace mode manually, the computer must be in the stopped condition and the fact should be ascertained that the job program for which trace mode is to be set is currently in control. Failure to comply can cause termination of the program in control when the trace mode is set (either a different job program or EXEC). Again, the dynamic method is suggested.
- 3) When the trace mode is set and a jump is performed, the central processor automatically loads the lower half of film memory location 103 with the address of the location following the jump instruction except in the case of a modifier jump (MOJP). For the modifier jump, this half word is cleared to zero.
- 4) The EXEC subtracts one from the captured P-setting and stores this value at the address in \$ERROR + 2 (address to the trace mode interrupt subroutine of the job program). This value is the address to which control would have gone if the trace mode interrupt had not occurred.
- 5) Occurrence of a trace mode interrupt terminates the trace mode. Trace mode can also be terminated by depressing the CL TRACE indicator. This latter operation is effective while the computer is running, but in the concurrent processing environment of EXEC, no assurance is given as to which program the trace mode is terminated for.
- 6) Memory lockout interrupts take precedence over trace interrupts. Hence, violation of memory lockout directs control to location 303 (memory interrupt) and execution of the jump instruction at this location will cause a trace interrupt.

VII. INPUT/OUTPUT

The Executive System accepts requests for the performance of I/O operations from the job programs operating under its control. EXEC must maintain common control of all I/O activities so that I/O operations of concurrently operating programs do not interfere with each other. Direct use of I/O instructions is illegal for programs operating under Executive control.

To request an I/O operation, each job program loads a fixed B-register with the address of an I/O Execution Packet and then enters the I/O section of the Executive System with a Load Modifier and Jump (LMJP) instruction. The Execution Packet is a list of words stored within the job program which specifies the operation to be performed, the logical channel, and the units involved.

When an I/O request is made, the channel on which operation is desired is either busy or not in use.

If a request is submitted when the required channel is not busy, the requested operation is initiated immediately. When a request is received and the specified channel is busy servicing a previously-submitted request, EXEC associates the address of the newly submitted Execution Packet with the identity of the requesting program and stores these items in one of three priority lists associated with each channel. Whether or not the channel is busy, control is normally returned to the requesting program after the I/O request is initiated or stored. See Appendix J for UNISERVO IIIA dual-channel operation.

The job program specifies in which of the three lists the Execution Packet address is to be stored, should storing be necessary. These lists are known as the High-Priority Request List, the Medium Priority Request List, and the Low Priority Request List, respectively. There are three such priority lists for each of the 15 allocatable channels (i.e., not including the console channel) making a total of 45 in all. See Appendix J for UNISERVO IIIA dual-channel operation.

If, during processing of interrupts, it is determined that all work specified by an Execution Packet has been completed, a search is made of the High Priority Request List, Medium Priority Request List, and Low Priority Request List, in that order, to obtain the address of a new Execution Packet. Within each priority group requests are taken, without regard to the identity of the program which entered them, in the same order as that in which they were entered in the priority group.

A. Submission of Requests

Requests for execution of I/O operations are submitted to EXEC via the following calling sequence:

8	9	FUNCTION	14	15	SUB FIELDS	
		L	D	P	\$Q α , a :	
		L	M	J	P	\$B1, \$X10 :

where \$X10 is the entrance to the I/O section of the Executive System, and

a is the address of a Request Parameter.

The Request Parameter at address a has the form:

8	9	FUNCTION	14	15	SUB FIELDS
			H		p, e :

where e is the address of the first word of the associated I/O Execution Packet, and

p denotes the request list priority assignment.

The parameter p may take on values from 0 to 3 as follows:

- When p = 0, the request is listed in the Low Priority Request List.
- p = 1, the request is listed in the Medium Priority Request List.
- p = 2, the request is listed in the High-Priority Request List.
- p = 3, a search is made of the Low Priority Request List for a request identified by address e, which was previously submitted by the program currently in control. If such a request is found, it is removed from the Low Priority Request List and placed in the Medium Priority Request List at a logical position such that it will be used after all requests previously entered in the Medium Priority Request List. No error indication is returned to the requesting program if the request whose priority is to be upgraded is not found in the Low Priority Request List.

The priority specification does not apply if the request can be serviced immediately. All values of p other than 0-3 are illegal for job programs.

There are occasions when requests are temporarily bypassed in favor of requests placed lower in the list. For example, if a tape unit is rewinding, it may be possible to service a request further down the list for a unit which is not rewinding. With respect to a given peripheral unit, I/O requests are always serviced in order of their position on the list: priority 2, position 1, 2, 3, ..., priority 1, position 1, 2, ..., etc. With respect to a given program, drum requests are always serviced in order of their position on the list per drum channel.

Interrupts which occur during intermediate phases of request processing will signal EXEC to issue I/O commands to read or write the next block, line or card. If an error is indicated, it is logged in a Summary Table which can be transferred to an external medium, and error recovery is attempted. If manual intervention is required, that fact will be made known to the operator by means of the console typewriter, along with the channel and unit involved. He will reply with a type-in indicating whether or not continued operation on the subject peripheral unit is feasible. After interrupt processing, if more work remains to be done on an Execution Packet current for the interrupting channel, control is returned to whatever program was operating when the interrupt occurred. Under these conditions, the Executive System saves and restores only the film memory registers it needs to process the interrupt. Whenever a request packet is terminated, the Switching section of EXEC may switch control to another program, according to the procedure described in Section VIII.

B. I/O Execution Packet: General

The format of an I/O Execution Packet is illustrated in Figure 9. Depending on the operation to be performed, a packet may contain from two to six words. The first two words shown, the STATUS WORD and FUNCTION WORD must always be present. The other words, if applicable, must appear in the order shown and in contiguous memory locations.

This implies that words which are not applicable in the I/O Execution Packet for a particular function are not to be included.

When a packet is submitted for execution and the channel is busy, the request is entered into the request list. If the channel is not busy when the packet is submitted, the packet is checked for legality and the status code is changed to "request in progress." The facility table in the Executive System is compared with the facilities requested in the packet to make sure there is no overlap with other programs. An image of the packet is stored in EXEC so that it cannot be accidentally overlaid with erroneous information while the request is in progress. However, the buffer status word and block read are updated at both locations when a block, card, or line has been successfully processed.

When a request packet is terminated because of an end-of-file, stop sentinel, non-recoverable equipment error, or any other conditions under which it would be undesirable to process additional requests for operation on a peripheral unit, EXEC sets a "logical interlock" indicator applying to the unit involved.¹⁵ After this indicator is set, EXEC terminates requests for operation on the interlocked unit, either from the request lists or those which are submitted after the interlock has been set (provided the channel is not busy when the request is submitted), until a request to "remove logical interlock" is received. Logical interlock does not apply to drum channels.

The words which form the I/O execution packet are described in the following subsections.

1. Status Word

By inspection of the status word a job program determines whether the related I/O request is stored in the waiting list, being serviced, completed normally, or terminated because of a contingency. Before the Executive System returns control to a program which has executed the request submission calling sequence, any former content of bits 35-30 is replaced by a new status code. The status codes are as follows. The conditions associated with codes 20-34 causes the program to be terminated.

¹⁵ The word "termination" here refers to the deletion of the request from a request list either because of normal completion or because of an error condition from which recovery is not possible.

<u>Octal Code</u>	<u>Meaning</u>
ØØ	The request has been completed normally. If there were any errors, recovery was successful.
Ø1	Logical interlock was in effect at the time request was to be executed.
Ø2	A "find" was made in a "read with sentinel check" operation. Logical interlock has been set.
Ø3	End-of-file (magnetic tape), invalid address during execution of function (drum), or an attempt was made to read a UNISERVO IIA tape backwards when it was in the rewound position. Logical interlock has been set for non-drum units. For a UNISERVO IIIC, a file separator was encountered during a forward read or search. If there is a block of data after the file separator, the tape is positioned ready to read that block. If the tape has not been recorded beyond the file separator, another forward read or search will result in a runaway tape. A backspace file operation will reposition the tape so that the tape mark is ahead of the read head, making it possible to back over the file with a backspace block operation. Status code 03 is also stored in response to backspace block if a tape mark is passed over.
Ø4	End of magnetic tape, end of paper tape, or end of card deck. Logical interlock has been set.
Ø5	End-of-block detected during a block search drum or block search read drum operation, i.e., a "find" was not made. For UNISERVO IIIC operation, an abnormal frame count error status code was received while reading the block just passed over; i.e., an exact multiple of six characters was not read. Due to the IBM recording formats it may or may not indicate a read error. S6 of the last word read into core has a count of the number of significant characters in the last input word.

Octal CodeMeaning

06	A nonrecoverable error has occurred in reading the word following the drum end-of-block word. If cards were being read, an illegal character was found on the card most recently read. Logical interlock has been set in the latter case.
07	Nonrecoverable drum parity error. The address of the word in error is in bits 22-0 of the status word.
10	Not used.
11	A nonrecoverable read or write error has occurred, as specified by the interrupt code in bits 29-24 of the packet status word. Logical interlock is set for non-drum units. For block functions on magnetic tape, bits 35-18 of the record count word contain the number of blocks read or moved since the function was begun, including the erroneous block. The buffer status word reflects the number of words transferred into memory including those taken from the erroneous block.
12	Not used.
13	A read or write error on magnetic tape, card unit, or printer has resulted in an indeterminate position of the input or output medium. Logical interlock is set.
14	During the execution of the request, an interrupt indicating manual intervention was received. An EXEC I/O error message was typed out, to which the operator responded that the requested unit was "down". Logical interlock is set for nondrum units.
15	The requested (nondrum) unit was declared "down" before the request was serviced.
16	While this request was in the request list, it was cancelled by a "terminate" request.

<u>Octal Code</u>	<u>Meaning</u>
17	While the request was being executed, it was halted by a "terminate" request.
20	A priority specification other than 0, 1, 2, or 3 was in bits 35-18 of \$Q0.
21	The function code was not defined for the specified channel.
22	The master bit unit designator field, 15-0 of Word (2), contained no master bits, more than one master bit, specified a unit not part of the EXEC environment; or specified an input-only or output-only unit not consistent with the function code.
23	The specified unit was not assigned to the requesting program.
24	One or more words in the request packet, other than the first two, overlap a core area not assigned to the requesting program. The function code is used to determine the number of words which constitute a request packet.
25	Illegal input buffer starting address, or too many input words requested (cards).
26	The drum area to which writing is specified, includes locations assigned to another program.
27	An attempt was made to initiate a write on a magnetic tape unit, with a zero word count, as specified in bits 33-18 of (3).
30	Invalid starting address requested in drum read or search.
31	One or both of the first two words in the request packet overlap a core area not assigned to this program.
32	More than 65,535 output words requested in card punch operation.
33	Read requested from non-existent drum address. (Operator response of F to I/O Error message 22.)

Octal CodeMeaning

34	Write with BCD specified has been attempted on UNISERVO IIIC with S1 of the first word of the block equal to zero.
40	The execution packet address is stored in the request list.
41 - 77	The request is currently being serviced. If the execution packet contains fields indicating buffer status or number of blocks, lines, or cards processed, and if the requesting program has control, it may determine from these fields how many words of input or output have been processed so far.

It is possible to determine from the status code whether or not a request was terminated because of a logical interlock condition.

An interrupt code is contained in bit positions 29-24 to supplement the information contained in the status code. This is the external interrupt code most recently received from the channel through which the request was processed.

Upon termination of magnetic drum functions, bit positions 22-0 may contain a copy of the last 23 bits of the word following an End-of-Block word (see I/O Functions below), or the address of a drum search "Find" word. In case of a parity error, bit positions 22-0 will contain the address of the word causing the error.

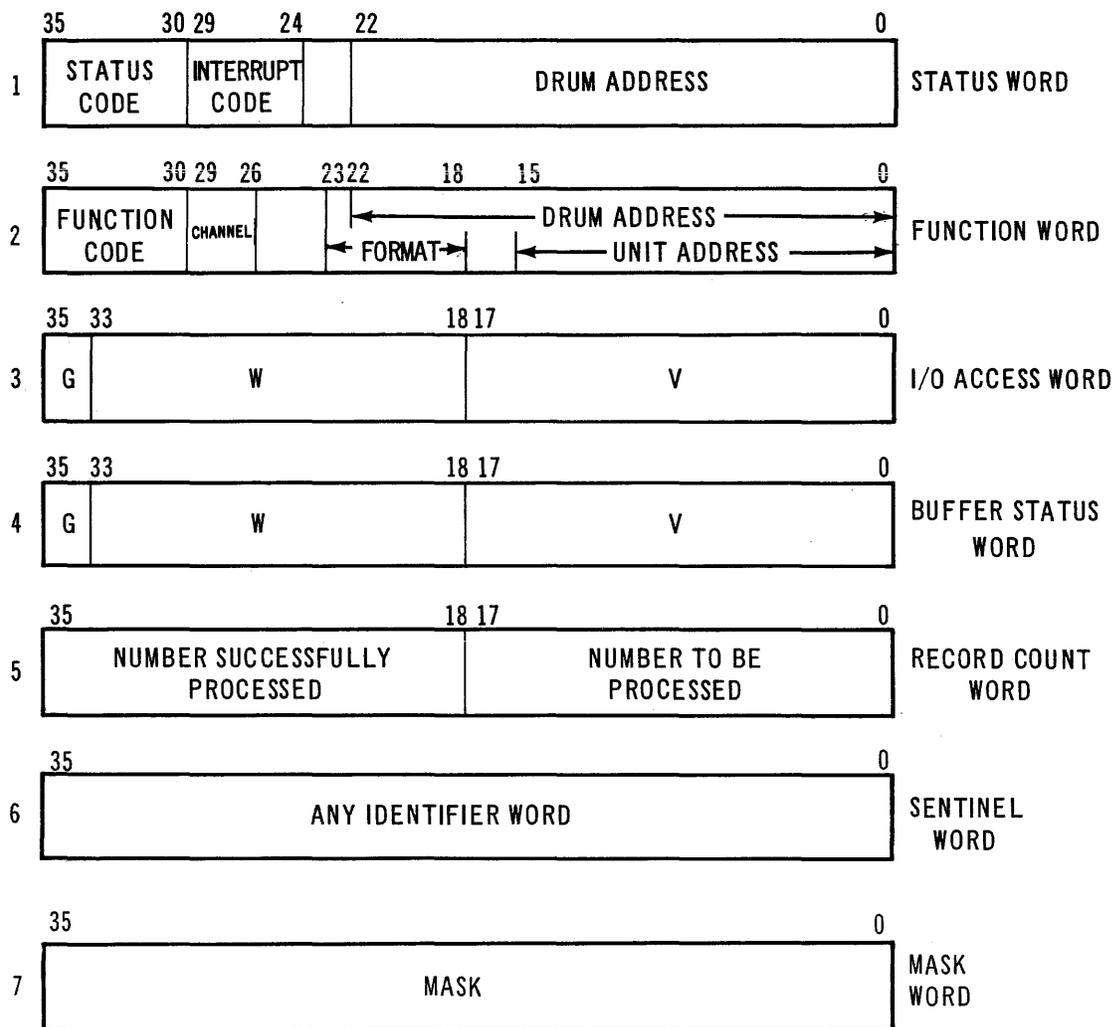


Figure 9. I/O Execution Packet

- FUNCTION WORD:** Bit positions 35-30 contain a 6-bit number specifying the operation to be performed. A 4-bit channel designator occupies positions 29-26. If a drum operation is called for, bit positions 22-0 specify a drum starting address. For all other operations, positions 23-18 may contain format information and positions 15-0 may specify (by Master Bit Selection, i.e., by position of a "one" within the binary field) the peripheral unit on which operation is desired. No portion of this word is altered by the Executive System. Positions 15-0 should be zero for card reader or punch operations.

3. I/O ACCESS WORD: Data transfer to and from core memory is controlled by the I/O ACCESS WORD. This word is entered into the appropriate film memory location by EXEC. Bit positions 17-0 contain the core memory address, V, at which the transfer is to begin. Positions 33-18, W, represents the number of words to be transferred, positions 35-34 form the Transfer Mode Designator, G, which can specify

increment V	(G=00)
inhibit increment V	(G=01)
decrement V	(G=10)
inhibit decrement V	(G=11)

EXEC does not check the validity of the V portion of the I/O access word for output requests. For input operations, V is checked to ascertain that transfer begins at an address assigned to the requesting program. In addition, when W is not supplied by EXEC, the G, W, and V designators are used to check that all locations to which input data may be transferred are assigned to the requesting program. If such is not the case, the original W designator is modified and input transfer will end just short of an area not assigned to the program. The modification of W is intended to permit reading program segments of unknown length from tape or drum.

The word count, W, is ignored in High-Speed Printer requests. An appropriate value of W is computed and stored by EXEC for card-read and card-punch requests.

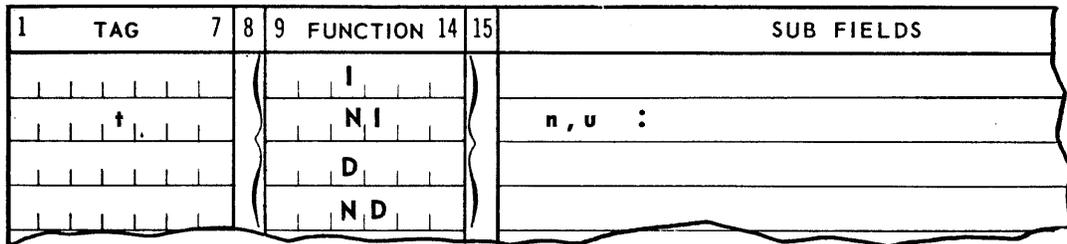
4. BUFFER STATUS WORD: While the I/O ACCESS WORD is controlling data transfer from its film memory location, the address of the current data transfer, V, is incremented or decremented if either is specified by the designator G. The number of words to be transferred, W, is decremented. Upon termination of each function, i.e., after each block, line, card, or group of words is successfully processed, the film memory word is stored intact in the BUFFER STATUS WORD location of the Execution Packet. When a non-recoverable error, or end-of-block condition, results in a transfer of fewer words than specified, the number of words actually read or written can be determined by inspecting the BUFFER STATUS WORD. Its contents are ignored when a request is submitted for execution.

where f is any one of the mnemonic codes recognized by SLEUTH as specifying an Executive I/O function code (see below).

- a is (1) an I/O Unit Tag, or
 - (2) a Drum Table Tag, or
 - (3) a Drum Table Tag \pm constant, or
 - (4) a Drum Table Tag \pm Drum Table Length Tag

x is an integer specifying format for those Executive I/O functions where a format specification is required. This field may, for example, specify the number of lines to be spaced (skipped) prior to printing a line on the High-Speed Printer.

WORD 3: This word is coded as an I/O Access Control Word



where I specifies increment
 NI specifies no increment
 D specifies decrement
 ND specifies no decrement

- n is (1) a constant, or
 - (2) a Data Table Length Tag, or

- (3) a Data Table Length Tag \pm constant
- u is
- (1) a label, or
 - (2) a label \pm constant, or
 - (3) a Data Table Tag, or
 - (4) a Data Table Tag \pm constant, or
 - (5) a Data Table Tag \pm Data Table Length Tag

WORD 4: This is coded as a whole word of zeros, as

1	TAG	7	8	9	FUNCTION	14	15	SUB FIELDS
t			W			ø :		

WORD 5: This word is coded as a half-word, as

1	TAG	7	8	9	FUNCTION	14	15	SUB FIELDS
t			H			ø, n :		

- where n is
- (1) a constant, or
 - (2) a tag, or
 - (3) a tag \pm constant

WORD 6: This may be coded using any form which will produce the desired 36-bit identifier, as, for example

1	TAG	7	8	9	FUNCTION	14	15	SUB FIELDS
t			W			i :		

or

1	TAG	7	8	9	FUNCTION	14	15	SUB FIELDS
t			S C			i, C ₁ , ..., C ₆ :		

- where i may be
- (1) a constant, or
 - (2) a tag, or
 - (3) a tag \pm constant

and

C₁...C₆ are any six Fielddata characters.

WORD 7: This may be in any form which will produce the desired 36-bit mask.

In addition to these words it is also necessary to code the proper declaratives to define the I/O Unit Tags, Drum Table Tags, and Drum Table Length Tags which are used in coding words 2 and 3.

The following examples illustrate the coding of some commonly used I/O Execution Packets:

1. Three 200-word blocks are to be read in the forward direction from a tape referred to symbolically as "INTAPE". These blocks are to be read into consecutively increasing addresses of a core area beginning at the address referred to as "INBUFF". The I/O Execution Packet may then be coded as

1	TAG	7	8	9	FUNCTION	14	15	SUB FIELDS	37
	R	E	A	D	T	P		Ø	:
					W			INTAPE	:
					R	T	F	N	:
					I			600, INBUFF	:
					W			Ø	:
					H			Ø, 3	:
									:

Note that only five words are required for this packet. The mnemonic function code "RTFN" indicates to the Executive I/O Functional Routines: "Read Tape Forward."

2. The contents of a drum table are to be read into consecutively increasing addresses of core memory such that the first word of the table read from drum is stored at symbolic core address "INBUFF". The drum table has been defined by a declarative as having symbolic name "DATA" and symbolic length "DATAL". The packet may be coded as

1	TAG	7	8	9	FUNCTION	14	15	SUB FIELDS	37
	R	D	D	R	U	M		Ø	:
					W			DATA	:
					R	D		DATAL, INBUFF	:
					I			Ø	:
					W				:

where the function code "RD" indicates "Read Drum."

3. Three lines are to be printed on the High-Speed Printer with 4 lines of spacing preceding each printed line. The first word to be printed is stored at symbolic core address "PTBUFF." The printer to be used has been defined symbolically by an I/O unit declarative as "PRINTR." The packet may be coded as

1	TAG	7	8	9	FUNCTION	14	15	SUB FIELDS	37
	P	R	I	N	T			Ø	:
								PRINTR, 4	:
								Ø, PTBUFF	:
								Ø	:
								Ø, 3	:

where the function code "PHSP" indicates "Print on High-Speed Printer."

4. A tape referred to symbolically as INTAPE is to be moved five blocks in the forward direction. The packet may be coded as

1	TAG	7	8	9	FUNCTION	14	15	SUB FIELDS	37
	M	O	V	E				Ø	:
								INTAPE	:
								Ø, 5	:

D. I/O Functions

The I/O Section of EXEC initiates appropriate function and I/O transfers to and from the peripheral equipment, as specified by an I/O Execution Packet in the program's data area. Requests for I/O operations are identified in the Request List by the addresses of the first word of the Execution Packets. A 6-bit function code in the FUNCTION WORD of the packet specifies the operation to be performed. Functions which may be specified correspond in general with commands which may be given to the Channel Synchronizers, with the following differences:

1. The initiation of Function Transfer, and of Input or Output Transfer is handled by the I/O section of EXEC rather than by each job program. EXEC also supplies any Output Access Words needed for function transfer.

2. Machine functions which are terminated with an error indication are automatically repeated or re-submitted in a modified form by the I/O section whenever such a procedure would provide a recovery from the error condition. Thus, recovery commands such as "Read Tape Backward at Low Gain" are not specified directly by the job programs but are issued by EXEC in case of reading difficulties. In certain cases, the recovery procedure may be a type-out to the operator to correct a mechanical condition, and an expected reply. In this case, the Execution Packet is temporarily retired until the operator replies. If the operator cannot correct the error condition, the request is terminated.
3. All EXEC functions for the UNISERVO IIA fixed-block mode have unique 6-bit codes as opposed to the machine function which makes use of the same codes in bits 35-30 but has a one in bit 20 for all fixed-block mode instructions in word 3 of the request packet. When writing in fixed block mode, the word count is set to 120 by EXEC. If a read is requested, the word count is set to an exact multiple of 120 depending upon the contents of bits 17-00 of word 5.
4. The UNISERVO IIIA function codes all have corresponding UNISERVO IIA function codes where applicable. See Table 6 for a complete listing of the functions.

Table 6 describes the Executive System I/O Function Repertoire. The column titled "EXECUTION PACKET WORDS" specifies which words of the I/O execution packet must be present for the particular function. It is important to note that the functions described here are not necessarily machine instructions, but are instead pseudo-functions (or system macros) which are used by the Executive to reference its own library of I/O Functional routines.

TABLE 6: EXECUTIVE I/O FUNCTION REPERTOIRE

FUNCTION	EXECUTION PACKET WORDS	EXPLANATION
UNISERVO IIA* and IIIA FUNCTIONS		
Read Tape Forward	1,2,3,4,5,	Magnetic tape on the specified unit is read forward the number of blocks specified by word 5 of the Execution Packet. (For all tape operations where word 5 applies to a packet, the I/O ACCESS WORD that applies to each block is the I/O ACCESS WORD as modified in accordance with data transfers occurring in the preceding block. If the number of words to be transferred is counted down to zero before the required number of blocks have been processed, there will be no data transfer when more blocks are processed.)
Read Tape Backward	1,2,3,4,5	Same as above except that tape movement occurs in the backward direction.
Search Read Tape Forward	1,2,3,4,5,6	<p>Tape on the specified unit is moved forward until a block is detected whose first word is identical to the identifier contained in word 6 of the Execution Packet. The I/O ACCESS WORD then controls input data transfer from the block containing the "find". Normally tape movement is concluded only when the number of blocks processed is equal to the number to be processed. Abnormal operations may cause termination before this condition is satisfied.</p> <p>NOTE: The UNISERVO IIIA hardware does not allow for reading the sentinel word into core on a search function. EXEC simulates this by transferring the sentinel word to the first word of the buffer when a find is made.</p>
Search Read Tape Backward	1,2,3,4,5,6	Tape is moved backward until a block is detected whose first word (in the backward direction) is identical to the SENTINEL WORD. The block containing the "find" is read in the backward direction with the I/O ACCESS WORD controlling the data transfer. Normally tape movement is concluded only when the number of blocks read is equal to the number to be read.

Read Tape Forward With Sentinel check 1,2,3,4,5,6	Magnetic tape on the specified unit is read forward the number of blocks indicated in word 5 of the Execution Packet, or until the first word of a block read is identical to the identifier in word 6. The number of blocks read will include the block containing the sentinel. A "logical interlock" is set for the tape unit referenced if the sentinel is detected.
Read Tape Backward With Sentinel check 1,2,3,4,5,6	Same as above except that tape movement occurs in the backward direction.
Move Tape Forward 1,2,5	Tape is moved forward the number of blocks specified in word 5 (third word in the packet).
Move Tape Backward 1,2,5	Tape is moved backward the number of blocks specified in word 5 (third word in the packet).
Rewind Tape 1,2	Rewind with interlock is initiated on the specified tape unit.
Rewind Tape With Interlock 1,2	Rewind with interlock is initiated on the specified tape unit.
Write Tape 1,2,3,4 12	One record is written on magnetic tape in the forward direction at 12.5 KC density. The I/O ACCESS WORD (word 3) determines the number of words in the record.
Write Tape 1,2,3,4 25	Same as above except that density is 25 KC

UNISERVO IIIA FUNCTIONS		
Masked Search Read Tape Forward	1,2,3,4 5,6,7	Tape on the specified unit is moved forward until a block is detected of which a portion of the first word, as defined by the MASK word, is identical to the SENTINEL word. The I/O ACCESS word then controls input data transfer from the block containing the find. Normally, tape movement is terminated only when the number of blocks read is equal to the number to be read.
Masked Search Read Tape Backward	1,2,3,4 5,6,7	Same as above except that tape movement is in the backward direction and a portion of the first word (as defined by the MASK word) of each block in the backward direction, is compared with the SENTINEL word.
Contingency Write	1,2	Hash which is skipped while reading is written on tape of the specified unit in even channels only for 2.5 inches until erased tape is being received at the write head.* The first block of valid data must be read successfully in the forward direction before a contingency write function is required.
Write End- of-File	1,2	Tape on the specified unit is erased for 2.5 inches. Write end-of-file must be used only after a successful write.
UNISERVO IIIC FUNCTIONS		
Write Binary at High Density	1,2,3,4	One block is written at 556 characters per inch (CPI) with odd parity. The address of the first word to be written is given by bits 17-0 of the third word of the I/O execution packet. Bits 33-18 must specify a non-zero number of words to be written, while bits 35-34 may call for increment (00), inhibit increment (01), decrement (10), or inhibit decrement (11) of the address of current transfer. When the operation is successfully completed, bits 33-18 of word 4 of the I/O execution packet reflect a word count of zero, and bits 17-0 contain the address of the next word in core to be written.

*This function must be used prior to over-writing old data on a tape if the last function performed on that tape was not a Write or a Rewind.

Write Binary at Low Density	1,2,3,4	Same as above except that characters are written at 200 CPI density.
Write BCD at High Density	1,2,3,4	One block is written at 556 CPI with even parity. The I/O execution packet is interpreted in a manner similar to that for Write Binary at High Density. If BCD tapes are to be read by IBM equipment, only the BCD characters listed in Appendix A should be written. Note especially that characters consisting of all "zero" bits are not defined. In BCD mode, the presence of these characters will result in nonrecoverable write errors. EXEC terminates programs which attempt to write data in which the first six BCD characters contain all zeros.
Write BCD at Low Density	1,2,3,4	Same as above except that characters are written at 200 CPI density.
Write End-of-File at High Density	1,2	An end-of-file erased gap is recorded for a distance of 3.7 inches followed by a tape mark character (octal 17) and its longitudinal check character. When the file separator is passed over as a result of a read, search, or backspace block operation, an end-of-file status code is reported. The last block recorded on a tape must be followed by a file separator to insure detection of the end of recorded area. An arbitrary number of files, each consisting of a group of blocks ending in a file separator, may be recorded on one tape.
Write End-of-File at Low Density	1,2	Same as above except that it is intended for binary or BCD tapes written at low density.
Read Binary at High Density	1,2,3,4,5	Tape on the unit specified by word 2 of the I/O execution packet is read for the number of blocks specified in bits 17-0 of word 5. Word 3 specifies in the usual way the starting address of input transfer, number of words to be transferred, and the transfer mode designator. Word 4 gives, on completion of the operation, the address from which the next transfer would have occurred as well as the number of words remaining to be transferred. This operation, as well as all the read and search operations described below, must be used only for tape recorded with corresponding density and parity.

Read Binary at Low Density	1,2,3,4,5	Same as above except that this operation is used for binary (odd parity) tapes written at 200 CPI.
Read BCD at High Density	1,2,3,4,5	Similar to above except for density and parity.
Read BCD at Low Density	1,2,3,4,5	Same as above except for density
Read Binary at High Density with Sentinel check	1,2,3,4,5,6	Similar to Read Binary at High Density except that reading is halted after reading of a block containing in the first word a bit configuration identical to word 6 of the I/O execution packet. If reading is halted because of a sentinel "find", words 4 and 5 indicate the status of the input buffer and the number of blocks read, including the "find" block.
Read Binary at Low Density with Sentinel Check	1,2,3,4,5,6	Same as above except that this operation is intended for binary tapes recorded at low density.
Read BCD at High Density with Sentinel Check	1,2,3,4,5,6	Similar to above except for input tape density and parity.
Read BCD at Low Density with Sentinel Check	1,2,3,4,5,6	Same as above except for density.
Search Read Binary at High Density	1,2,3,4,5,6	Tape is moved forward until an equal comparison is found between the identifier in word 6 of the packet and the first word of a block. When a "find" is made, input transfer begins as governed by word 3 of the packet, and continues for the number of blocks specified in 17-0 of word 5.
Search Read Binary at Low Density	1,2,3,4,5,6	Same as above except for density.
Search Read BCD at High Density	1,2,3,4,5,6	Similar to above except for density and parity.

Search Read BCD at Low Density	1,2,3, 4,5,6	Same as above except for density.
Backspace Block	1,2	Tape recorded at any density or parity is backspaced one block.
Backspace File	1,2	Tape is moved backward until a block of 4 or less frames has been passed over. The next forward read will cause an end-of-file to be reported, if the block was a valid tape mark block, and leave the tape positioned ready to read the first block in the file which was partially or completely passed over by the Backspace File operation. If, on the other hand, a Backspace File is followed by a Backspace Block operation, the read head will be positioned ready to read the last block of the file before the one partially or completely passed over by the Backspace File operation.
Rewind Tape	1,2	Tape is rewound to load point.
Rewind with Interlock	1,2	Tape is rewound to unload point and the transport is interlocked.
Skip while erasing	1,2	Tape is moved forward approximately 4 inches while writing all zeros.
MAGNETIC DRUM FUNCTIONS		
Read Drum	1,2,3,4	Starting with the address in bit positions 22-00 of word 2, data transfer continues until the number in positions 33-18 of word 3 is counted down to zero.
Block Read Drum	1,2,3, 4,5	The drum is read from the specified starting address. Data transfer is discontinued when the number of blocks specified by word 5, or the number of words specified by word 3, has been processed, whichever count is exhausted first. In the latter case, the count in H1 of word 5 is not incremented for the partial block read. (A block on drum is, by definition, ended by a word consisting of 36 "one" bits. This word is called the end-of-block word.) The count in bits 33-18 of the I/O access word should be high enough to allow input transfer of one or more words after the end-of-block word (all ones) if it is desired that EXEC store the "drum over-flow address" in bits 22-00 of the status word of the request packet.

Search on Drum	1,2,6	Starting with the address specified each drum word is compared with the SENTINEL WORD. If a "find" is made, the address of the "find" word is placed in positions 22-00 of word 1.
Block Search on Drum	1,2,6	Each drum word, from the specified starting address, is compared with the SENTINEL WORD. The search is discontinued when a find is made or when an end-of-block word is detected. The address of the "find" word, or the address in the overflow word in case of end-of-block, is placed in bit positions 22-00 of word 1.
Search Read Drum	1,2,3,4,6	Data transfer begins if a "find" word is detected and ends when the number of words specified by word 3 has been transferred.
Block Search Read	1,2,3,4,6	The drum is searched from the specified starting address. If a "find" is made, the I/O ACCESS WORD governs data transfer, starting with the "find" word. Data transfer ends when the end-of-block word is detected, or when the specified number of words is transferred whichever occurs first. The address in the overflow word in case of end-of-block is placed in bit positions 22-00 of word 1. The count in bits 33-18 of the I/O access word should be high enough to allow input transfer of one or more words after the end-of-block word (all ones) if it is desired that EXEC stores the "drum overflow address" in bits 22-00 of the status word of the request packet.

Chain Block Read Drum	1,2,3, 4,5	The drum is read from the specified starting address one block at a time using the contents of the overflow word as the address of the start of each successive block.* Data transfer is discontinued when the specified number of blocks (word 5) or number of words (word 3) has been read, whichever count is exhausted first. The count in bits 33-18 of the I/O access word should be high enough to allow input transfer of one or more words after the end-of-block word if it is desired that EXEC store the "drum overflow address" in bits 22-00 of the status word of the request packet.
Write Drum	1,2,3,4	The data area as defined by the I/O ACCESS WORD is copied on the drum, starting at the specified drum address. Any end-of-block words to be written must be in the data area.
PUNCHED CARD FUNCTIONS		
Condition for Field- data Input	1,2	Read and discard any data remaining in the card control unit memory and condition the control unit to translate input data from card code to Fielddata code.
Condition for Column- Binary Input	1,2	Read and discard any data remaining in the card control unit memory and condition the control unit to transfer input data "card image by column" to the central computer.
Condition for Row- Binary Input	1,2	Read and discard any data remaining in the card control unit memory and condition the control unit to transfer input data "card image by row" to the central computer.
Read One Card	1,2,3,4	One card image is transferred to the memory locations specified by word 3. A card is tripped only if the control unit buffer memory is empty.

*The overflow word follows the end-of-block word. This word is transferred by the peripheral subsystem to the external status word (octal address 311) of the core memory. The Executive System then picks up the least significant 23 bits of this word and transfers it to word 1 of the Execution Packet.

Read and Trip Fill	1,2,3,4,5	The number of cards to be read is specified by positions 17-00 of word 5. Bits 33-18 of word 3 are set to the correct word count according to the translation mode last set; i.e., the number of words per card for the current translation mode is multiplied by the number of cards to be read. The number of cards to be read is specified in bits 17-00 of word 5.
Read and Trip Fill with Sentinel Check	1,2,3,4,5,6	This function is the same as Read and Trip Fill with the following exception. When the first computer word of data from a card being read is identical to the identifier in word 6, reading is discontinued. A "logical interlock" is set for the referenced card reader if the sentinel is detected.
Condition for Fielddata Output	1,2	For succeeding cards, the card control unit is conditioned to receive Fielddata output for translation to card code.
Condition for Column Binary Output	1,2	For succeeding cards, the card control unit is conditioned to receive binary output to punch "by columns".
Condition for Row Binary Output	1,2	For succeeding cards, the card control unit is conditioned to receive binary output to punch "by row".
Punch 80-Column Cards, Select Stacker Zero	1,2,3,4,5	Bits 33-18 of word 3 are set to the correct word count by multiplying the number of cards requested by the number of words per card that will be transferred according to the translation mode last set. Bits 33-18 of word 4 are used for temporary storage by EXEC. The number of cards to be punched is specified in bit positions 17-00 of word 5.
Punch 80-Column Cards, Select Stacker One	1,2,3,4,5	Same as above with the following exception. Cards successfully punched as well as cards having verification errors will be sent to stacker one.

HIGH-SPEED PRINTER FUNCTIONS

<p>Print on High-Speed Printer</p>	<p>1,2,3,4,5</p>	<p>Groups of 22 computer words or less (containing Fielddata coded characters) starting at the address specified in positions 17-00 of word 3 are transferred to the High-Speed Printer until the number of lines printed is equal to the count specified in positions 17-00 of word 5. For this request, the number in positions 33-18 of word 3 is ignored. If a Fielddata stop code (octal 77) is used, only the words preceding and including the word containing the stop code will be transferred to the High-Speed Printer. Data transfer for the next line of print will begin following the word containing the stop code. Before each line is printed, the paper in the printer is spaced from zero to 63 spaces, as specified in positions 23-18 of word 2. The BUFFER STATUS WORD will contain an address one greater than the address from which the last data was transferred to the High-Speed Printer.</p>
------------------------------------	------------------	--

<p>Print on High-Speed Printer in Variable Format</p>	<p>1,2,3,4,5</p>	<p>This function is the same as Print on High-Speed Printer with the following exceptions. The line spacing (in octal) for each line to be printed is contained in the least significant character position of the line last printed. The buffer for the first line of print must contain an extra word (the first word) which contains the line spacing for that line. If stop codes are used, the line spacing must be in the same word as the stop code.</p>
---	------------------	---

PUNCHED PAPER TAPE FUNCTIONS

<p>Read Paper Tape Forward</p>	<p>1,2,3,4</p>	<p>Characters (frames) are read from paper tape in the forward direction and transferred to computer core memory as specified by word 3.</p>
--------------------------------	----------------	--

Read Paper Tape Backward	1,2,3,4	Same as above except tape movement is in the backward direction. Due to hardware design, it is not recommended to attempt to read more than 120 frames in the backward direction (tape must be manually wound).
Punch Paper Tape	1,2,3,4	Characters (frames) are punched in paper tape from the contents of computer core memory as specified by word 3.
REQUEST CONTROL FUNCTIONS		
Remove Logical Interlock	1,2	Upon receipt of this request, EXEC clears the logical interlock indicator associated with the specified nondrum unit and terminates any request for the unit which may still be in the request list. Requests for RLI are always serviced immediately; therefore, no "wait-for-completion" sequence is necessary. The RLI function should be used whenever a completion status code indicates that logical interlock is set. At the time logical interlock is set for a unit, a search is made through the request list for the channel to which the unit is assigned for the purpose of terminating all requests for the subject unit. The request for the unit for which logical interlock is set is given a status code of 01. When a remove logical interlock function is received, a second search is made of the request list for requests listed between the time the interlock is set and removed; the status of these requests are set to 01. Requests for RLI on drum channels or for units not in a logically interlocked condition are treated as legal "do-nothing" operations.

<p>Terminate Outstanding Requests</p>	<p>1,2</p>	<p>Request identified by word 1 and/or 2 are cleared from the requests list and terminated with a status code of 16 octal. If there is a request in progress, it is terminated after the current block, line, card, or group of words is transferred. A status code of 17 octal is used for in-progress requests terminated before the specified count has been fulfilled. As in the case of RLI, the TERM request is serviced immediately and is normally completed with a status code of 00. Peripheral units other than drums are identified as in other requests; i.e., by a channel number and single master bit unit designator in word 2. For drum channels, each request is terminated where the initial drum address, X, meets the condition $L \leq X \leq H$ where L is the number in positions 22-00 of word 1 of the "terminate" request and H is given by positions 22-00 of word 2.</p> <p>EXEC terminates outstanding requests for job programs which are closed out. To be terminated by any job program, a request must have been originally submitted by that program. The terminate request neither sets nor clears the logical interlock described above.</p>
---------------------------------------	------------	---

EXEC I/O Function List

Function	EXEC Mnemonic	EXEC Octal
UNISERVO IIA TAPE UNIT (variable block mode)		
Read Tape Forward	RTF	42
Read Tape Backward	RTB	62
Search Tape Forward	SRTF	46
Search Tape Backward	SRTB	66
Read Tape Forward with Sentinel Check	RTFS	43
Read Tape Backward with Sentinel Check	RTBS	63
Move Tape Forward	MTF	41
Move Tape Backward	MTB	61
Rewind Tape	REW	20
Rewind with Interlock	REWL	21
Write Tape at 12.5 kc	WTL	01
Write Tape at 25. kc	WTH	02
UNISERVO IIA TAPE UNIT (fixed block mode)		
Read Tape Forward	FRTF	52
Read Tape Backward	FRTB	72
Search Tape Forward	FSTF	56
Search Tape Backward	FSTB	76
Read Forward with Sentinel Check	FRFS	53
Read Backward with Sentinel Check	FRBS	73
Move Tape Forward	FMTF	51
Move Tape Backward	FMTB	71
Write Tape at Low Density	FWTL	03
Write Tape at High Density	FWTH	04
Rewind Tape	REW	20
Rewind Tape with Interlock	REWL	21
UNISERVO IIIA TAPE UNIT		
Read Forward	RTF	42
Read Backward	RTB	62
Search Forward	SRTF	46
Search Backward	SRTB	66
Read Forward with Sentinel Check	RTFS	43
Read Backward with Sentinel Check	RTBS	63
Move Forward	MTF	41
Move Backward	MTB	61
Rewind	REW	20
Rewind with Interlock	REWL	21
Write Tape	WTH	02
Masked Search Forward	MSF	47
Masked Search Backward	MSB	67
Contingency Write	CW	03
Write End-of-File	WEFH	04

EXEC I/O Function List (cont.)

Function	EXEC Mnemonic	EXEC Octal
UNISERVO IIIC TAPE UNIT		
Write Binary at High Density	WBH	02
Write Binary at Low Density	WBL	01
Write BCD at High Density	WDH	06
Write BCD at Low Density	WDL	05
Write End-of-File at High Density	WEFH	04
Write End-of-File at Low Density	WEFL	03
Read Binary at High Density	RBH	42
Read Binary at Low Density	RBL	37
Read BCD at High Density	RDH	52
Read BCD at Low Density	RDL	47
Read Binary High with Sentinel Check	RBHS	43
Read Binary Low with Sentinel Check	RBLS	40
Read BCD High with Sentinel Check	RDHS	53
Read BCD Low with Sentinel Check	RDLS	50
Search Binary at High Density	SRBH	46
Search Binary at Low Density	SRBL	44
Search BCD at High Density	SRDH	56
Search BCD at Low Density	SRDL	54
Backspace Block	BSB	61
Backspace File	BSF	64
Rewind	REW	20
Rewind with Interlock	REWL	21
Skip While Erasing	SKIP	10
Magnetic Drum		
Read Drum	RD	42
Block Read Drum	BRD	52
Search Drum	SD	45
Block Search Drum	BSD	55
Search Read Drum	SRD	46
Block Search Read Drum	BSRD	56
Chain Block Read Drum	CBRD	62
Write Drum	WD	02
Punched Card		
Condition Fielddata Input	CFDI	62
Condition Column Binary Input	CCBI	63
Condition Row Binary Input	CRBI	64
Read Card	RC	41
Read Card Trip Fill	RCTF	42
Read Card Trip Fill with Sentinel Check	RCTS	44
Condition Fielddata Output	CFDO	04
Condition Column Binary Output	CCBO	05
Condition Row Binary Output	CRBO	06
Punch Card Stacker Ø	PCSØ	02
Punch Card Stacker 1	PCS1	03

EXEC I/O Function List (cont.)

Function	EXEC Mnemonic	EXEC Octal
High-Speed Printer		
Print HSP Print HSP Variable Format	PHSP PHSV	02 03
Paper Tape		
Read Paper Tape Forward Read Paper Tape Backward Punch Paper Tape	RPT RPB PPT	42 62 02
Control		
Remove Logical Interlock Terminate	RLI TERM	07 23

E. I/O ERRORS AND RECOVERY

Unusual I/O Interrupt conditions which the operator should know about are communicated on the typewriter in the form of I/O error typeouts. I/O error messages are in the format:

I/O ERR *mm* -Ccc-Uuu-Du-Aaaaaaaaa-Fff-Ffff-Gff-Xii-Txxyzz-b

b=N706024
N706054
N004445

All fields which could appear are shown above: usually only 3 or 4 fields are present in an I/O error typeout. If the message originates from the I/O Error Logging, ERR is replaced with LOG (see Section 5 below).

mm - messagenumber indicative of the conditions of the error (see Table 7)
Ccc - channel (decimal)
Fff - most recent external function transferred (Ffff for UNISERVO IIIC)
Gff - nonsense interrupt (message #15) type code (Gfff for IIIC)
Xii - interrupt code

ii = 01 input monitor
ii = 02 output monitor
ii = 03 function monitor
all other = except interrupt code

Txxyzz - error count for the tape which has just rewound on the channel and unit shown.

xx = 70 EI counts - IIA, IIIA
yy = 60 EI counts - IIA, IIIA
yy = 44 (READ) EI - IIIC
zz = 24 EI counts (IIA)
zz = 54 EI counts (IIIA) - write only
zz = 44 EI counts (IIIC) - write only

N706024-IIA The interrupt codes shown in the message have
N706054-IIIA - reached the limit for the tape UNIT shown.
N004445-IIIC - sage format is for the read error count, and the 45 is for the write error count.

Uuu - unit number (decimal)
Du - drum unit number
Aaaaaaaaa - drum address (octal)

1. Special I/O Error Message

Message number 40 is a special message sent to the operator if no interrupt has been returned for the function shown on the channel and unit shown.

Each function sent out by EXEC has a unique time associated with it. This time is long enough to allow normal completion to take place.

The message is preceded by an arrow (Λ) to bring it to the operator's immediate attention.

The following action should be taken:

- 1) call a maintenance man
- 2) after the status of the channel is checked, the channel must be cleared
- 3) respond to the message with A, B, D, or E for all except drum channels. For drum use only an A or D response.

NOTE: The function may have been performed so that a response of A (try again) would result in the function being performed twice without the program being aware of this fact. This is particularly hazardous where physical movement of the I/O equipment is involved such as a write on tape in which case two identical blocks would be written.

It is not necessary to send an interrupt back to EXEC as the message response will simulate one.

No function will be initiated on the channel until the message is answered.

If the interrupt comes back after message 40 has been sent, it will be treated as a nonsense interrupt (message 15).

2. I/O Error Typeouts Requiring a Reply

Usually, a typeout requiring a reply is an indication that some sort of operator intervention is needed. The suggested actions below are grouped by type of equipment and message number. See Table 7.

TABLE 7.. ABBREVIATED EXPLANATION OF EXECUTIVE I/O MESSAGES

Errors Producing Message	Message	Legal Response
MT-60,70,20,54(IIIA),44(IIIC); Card-70	I/O ERR*01*-Ccc-Uuu-Fff-Xii	None
MT-24;UDI;HSP-UDI; MT-lost position	I/O ERR*02*-Ccc-Uuu-Fff-Xii	None
DRUM-06	I/O ERR*03*-Ccc-Aaaaaaaaa-Fff-Xii	None
DRUM-07,64	I/O ERR*04*-Ccc-Aaaaaaaaa-Fff-Xii	None
DRUM-20,60,70	I/O ERR*05*-Ccc-Du-Fff-Xii	None
MT-20,30,50,34(IIIC)	I/O LOG*06*-Ccc-Uuu-Xii	None
DRUM-60-70	I/O LOG*07*-Ccc-Du-Xii	None
DRUM-06,07,64 (Parity)	I/O LOG*10*-Ccc-Aaaaaaaaa	None
CHANNEL-20,30,50	I/O LOG*11*-Ccc-Xii	None
MT(REWIND)IIA	I/O LOG*12*-Ccc-Uuu-Txxyyzz-N7Ø6Ø24	None
MT(REWIND)IIIC	I/O LOG*13*-Ccc-Uuu-Tyyzz-NØØ4445	None
MT(REWIND)IIIA	I/O LOG*14*-Ccc-Uuu-Txxyyzz-N7Ø6Ø54	None
NSI	I/O ERR*15*-Ccc-Gff-Xii	None
MT-50,54,74,34(IIIC),UDI;CARD-50,74,UDI,20	I/O ERR*20*-Ccc-Uuu-Fff-Xii	A,B,D,E,C
HSP-50-54,74,UDI; MT54 (IIIA)	I/O ERR*21*-Ccc-Uuu-Fff-Xii	A,B,D
DRUM-14,50,54,UDI	I/O ERR*22*-Ccc-Du-Fff-Xii	A,D,F*
CARD54(PUNCH)30;MT-30	I/O ERR*23*-Ccc-Uuu-Fff-Xii	B,D
CARD-54 (READ)	I/O ERR*24*-Ccc-Uuu-Fff-Xii	A,B,C,D
DRUM-30	I/O ERR*25*-Ccc-Du-Fff-Xii	D,G
CARDS-60 (inappropriate function)	I/O ERR*26*-Ccc-Uuu-Fff-Xii	A,D
PT-ANY INTERRUPT	I/O ERR*30*-Ccc-Uuu-Xii	B,D,E
DRUM-ADDR.TABLE OVERFLOW WARNING	I/O ERR*31*-Ccc	None
MT-EI5Ø-IIIC, EI6Ø-IIIA,LATE OUTPUT ACKN.	I/O ERR*32*-Ccc-Uuu-Fff	A,D
MT,CARDS,PT,HSP,(no interrupt returned)	^I/O ERR*4Ø*-Ccc-Uuu-Fff	A,B,D,E
DRUM(no interrupts returned)	^I/O ERR*4Ø*-Ccc-Du-Fff	A,B,D,E

Key 1. Capitals in Message are Fixed Characters

2. Variable Characters
 Ccc - channel
 Fff - Function (if IIIC Channel-Ffff)

 Xii - external interrupt
 Uuu - Unit
 Du - drum unit
 Aaaaaaaaa - drum address
 Txxyyzz - See explanation-this page
 Gff - Nonsense interrupt type code,

3. Definitions (unexpected for)
 UDI - undefined interrupt given function
 NSI - nonsense interrupt (no funct. issued)

Note 1. N variable in Message 14 may not be present.
 (See complete writeup)

Note 2. Illegal Response will produce Message again preceded by two '?'s

*If illegal unit is shown it is program fault but, if legal unit it is machine trouble.

Response

- A-try again
 B-bad position
 C-cards-position good, unrecoverable error (or overpunch)
 D-equipment down
 E-end of cards,tape,etc.
 *F-Illegal address-Program Fault
 G-good position,unrecoverable error

I Field Explanation

- xx-7Ø EI Counts -IIA, IIIA
 yy-6Ø EI Counts -IIA, IIIA
 -44 EI Counts -IIIC (Read Only)
 zz-24 EI Counts -IIA
 -54 EI Counts -IIIA (Write Only)
 -44 EI Counts -IIIC (Write Only)

a. Magnetic Tape Channel

20 - Usually caused by interlock fault (EI 74). If the interlock can be corrected, reply A. Consult programmer instructions if the fault is due to trying to write on a reel having a master tape ring, or to operate on a unit that has been rewound with interlock. If programmer instructions do not cover these cases, reply E. Also use E if power drops due to the end of tape switch being activated, after restoring power and initiating a manual rewind. B should be used if it is suspected that power dropped in the middle of a tape operation (except for end of tape). If there is no known reason for the error interrupt, try A several times before resorting to D.

23 - B should be used the first one or two times this interrupt occurs, then D.

26 - Typed out when an abnormally long time is required to clear function mode when transferring an output function. Try A several times before using D.

b. High-Speed Printer Channel

21 - Correct interlock fault if possible and use response A. When reloading paper, the first new page should be positioned similarly to the page on which printing halted, if the printer ran out of paper in the middle of a run. If the printer was halted because of torn paper, reload paper and use B.

26 - Attempt to recover, using A.

c. Drum Channel

22 - If the external interrupt is 54 and the unit number typed out represents a drum unit present in the installation, try again. If the unit mentioned in the typeout does not exist, use F to inform the program about an illegal read.

25 - Reply G.

27 - Try again several times.

d. Card Channel

20

23 - From the external interrupt code and the indicator lights on the reader, punch, and control units, determine whether the error is a misfeed, jam, verification error, etc. and follow the instruction described under

24 Special I/O Procedures.

On only two occasions is it proper to clear a card channel by the Channel Clear button.

- (a) Before the Initial Load of the Executive is accomplished; and
- (b) During the process of rewriting the card control unit translation memory.

e. Paper Tape Channel

3Ø - If the equipment is down, use response D. If the reader or punch was halted because it had detected the end of tape, clear the tape fault light on the control unit operators panel and use E. If the cause of the error was other than those listed above, clear the fault from the control unit and respond with B.

3. Special I/O Procedures

TO RUN IN THE CARD READER - Turn power on. Load Cards. Depress the READ INTERLOCK switch on the control unit front panel, whether it is lit or not, to clear the read memory. Depress RUN on the card reader.

At installations where the convention prevails that (1) all programs use one of the conditioning instructions before reading a deck, and (2) all programs use a Read With Sentinel Check to recognize and stop at an "end-of-deck" sentinel, then the following procedure is convenient: at the end of each deck three blank cards are placed after the sentinel card. At the conclusion of the read there will be three blank card images in the card control unit memory and the read interlock will not be set. When the same or a different program communicates to the operator that a new deck should be loaded, it is only necessary to place the cards in the input hopper and respond Y to the communication; the READ INTLK or RUN buttons need not be depressed. The card images in the card control unit memory are cleared out when the conditioning request is executed.

TO UNLOAD CARD READER - Simply unload and repack cards. If the runin procedure described above is followed, no "run-out" is necessary. Respond Y to the unload or change typeout.

TO TEMPORARILY INTERRUPT READING - Depress STOP on the card reader. Depress RUN to start up again. In concurrent reading and punching, punching will also be temporarily interrupted.

READER MISFEED RECOVERY - If there are no more cards, reply E to inform the program of that fact. Otherwise dress up the cards in the input hopper if necessary. Depress RESUME (this will extinguish the READ INTL indicator). Reply A to the EXEC typeout announcing the misfeed.

READ VERIFICATION ERROR - Remove cards from the error hopper, replace them at the bottom of the input hopper. Depress READ CHECK on the control unit front panel. Reply A to the EXEC typeout announcing the verification error. If the read error persists after three successive tries, use the following procedure to inform the program that the card cannot be read: place the card which is at the bottom of the error hopper, on top of the output stack. Place the other two cards on the bottom of the input hopper. Depress the READ CHECK indicator switch. Reply C to the EXEC typeout announcing the unrecoverable read-check error.

TO RUN IN THE PUNCH - Turn power on. Load blank cards in the input hopper. Depress the PUNCH INTLK indicator switch on the control unit front panel. Depress RUN on the punch unit. The first card will be positioned in the punch station.

TO RUN OUT THE PUNCH - Remove remaining cards from the input hopper. Depress the END OF FILE switch so that the indicator lights up. Depress RESUME on the punch unit as many times as required to clear the punch. If the PUNCH CHECK indicator lights, attach a note to the punch deck indicating that one or more of the last three cards has a punch check error. After the punch is cleared, release the END OF FILE indicator switch.

Note: For a set of runs which closely follow each other, if it is known that a four-card run-out is performed under program control, the manual run-in and run-out may be omitted.

TO TEMPORARILY INTERRUPT PUNCHING - Depress STOP on the card punch. Depress RUN to start up again. In concurrent reading and punching, reading will also be temporarily interrupted.

PUNCH MISFEED RECOVERY - Dress up cards in input hopper if necessary. Depress RESUME. Reply A to the typeout announcing the misfeed.

CHIP BOX FULL-PUNCH - Remove, empty, and replace the chip box. Depress RESUME. Reply A.

PUNCH VERIFICATION ERROR - Reply D to inform the program that the card punch is down. Call maintenance.

CARD JAMS - READER AND PUNCH - If possible, remove the cards which caused the jam, depress the READ INTLK or PUNCH INTLK indicator switch, reset the JAM indicator (if jam in card reader) and reply B to the EXEC typeout.

If a maintenance enigneer is required to get the equipment back into operation, the typeout may be answered with D, without having the jam cleared, to signify that the reader or punch is "down".

BUFFER PARITY FAULT, READER AND PUNCH - Respond D (equipment down) to typeouts concerning reader and/or punch. The control unit memory must be rewritten before programs may again use card equipment.

OTHER ERRORS, READER AND PUNCH - If the error is an interlock due to a mechanical condition such as output bin full, power off, punch off, punch open, et., correct the cause of the interlock, depress RESUME, and reply A. In the case of an external interrupt code of 5Ø, 2Ø, or a code that is not defined, type in D (equipment down).

4. External Interrupt Codes and Messages

Table 7 is an abbreviated explanation of the messages which can come from EXEC I/O because of external interrupts. Messages are of two types: response and no response. For each response type of message, the abbreviated explanation shows which responses are legal to type back into EXEC. Each response produces a different status code.

No-response messages are for operator information only and are fixed as to the status code which will be returned to the working program.

The external interrupt codes displayed by I/O error messages are:

Magnetic Tape

- 2Ø - Sequence error; CS
- 24 - Sequence error (two block read); CU
- 3Ø - Character-count; CS
- 5Ø - Illegal function (34 Code on IIIC)
- 54 - Tape unit select error; Write error (IIIA)
- 6Ø - Parity error; CU
- 7Ø - Character-count; CU
- 74 - Interlock
- 44 - Write error (IIIC)

Magnetic Drum

- Ø6 - Parity error in overflow address
- Ø7 - Parity error not during continuous read
- 14 - Write fault
- 2Ø - CS sequence error
- 3Ø - CS character-count error
- 5Ø - Illegal function
- 6Ø - CU sequence error
- 64 - Parity error during continuous read
- 7Ø - CU character-count error
- 54 - Illegal address - either program or machine produced

Cards

- 2Ø - CS sequence error
- 3Ø - CS character-count error
- 5Ø - Illegal function
- 54 - Read/punch verification error
- 7Ø - Illegal character
- 74 - Interlock

High Speed Printer

- 5Ø - Illegal Function
- 54 - Unit Select error
- 74 - Interlock

Paper Tape

- Ø1 - Punch compare error
- Ø2 - End of paper tape or tape motion error

5. Messages From I/O Error Logging

Messages numbers 6 through 14 and 31 in table 7 originate from the I/O Error Logging routine. These messages are defined as follows:

a. Messages 6, 7, 11

The count for the unit shown for the interrupt shown has reached its limit. No further incrementation takes place until the logging tables are cleared as a result of the DER message.

b. Message 10

The parity error count for the drum address shown has reached a limit of 64. Incrementation will continue until a count of 2048 is reached.

c. Message 31

The table for storage of drum addresses at which parity errors have occurred is in danger of filling. The DER message should be used.

d. Message 12, 13, 14

The error count is shown for a tape on a unit just re-wound. See Section E above for an explanation of the I and N fields.

VIII. SWITCHING

A. The Dispatcher

The Dispatcher is a part of the Switching section of the Executive System. It accomplishes the switching of control among the various programs operating concurrently. To accomplish this, two Switch Lists are maintained to accommodate programs of two general types:

1. Switch List 1 controls those programs which can be assumed to voluntarily release control when they cannot continue their internal processing pending completion of a requested I/O function. It is further expected that programs in this list will voluntarily release control at relatively frequent intervals.
2. Switch List 2 controls those programs which cannot be expected to release control at frequent intervals. Control will be taken away from these programs when an I/O interrupt occurs signifying completion of a requested I/O function. These programs may also voluntarily release control. In either of these cases, control will be switched to the next program in Switch List 1 which is not in a "wait" condition, i.e., which is not awaiting completion of an I/O or Communication request. If all programs in Switch List 1 are in a "wait" condition, control will be transferred to the next program in Switch List 2.

The Dispatcher is referenced by the various sections of the Executive System to return control in the normal manner to the program which was operating at the time the Executive obtained control. These references will imply that the Dispatcher restore any film memory registers utilized by the Executive System to their condition at the time of interrupt, record the current clock register setting, and return control to the address at which the program was interrupted. A reference of this type is made by the I/O section of the Executive when it has received and listed an I/O request from an operating program, and when the Request List is not in an overflow condition (see Section VII).

When an I/O or communication request is received which places the Request List in an overflow condition, the Dispatcher takes control away from the requesting program. This means that the Dispatcher must save the contents of the film memory registers for the program and flag that program's switch list entry to indicate that the registers must be restored when the program is again referenced. The Dispatcher retains control until sufficient requests have been completed to alleviate the overflow condition.

When the I/O section of EXEC detects an interrupt indicating the completion of a requested I/O function, the Dispatcher is referenced to return control to a program in Switch List 1. If the interrupted program is from Switch List 1, control is returned to this program in the normal manner. If the interrupted program is from Switch List 2, then control is taken from this program and switched to a program in List 1. If no program in List 1 can operate, control is switched to the next program in List 2. In the latter two cases, the Dispatcher saves the contents of the film memory registers for the program from which control is taken and flags its switch list entry accordingly.

It is intended that Switch List 1 will cycle at a faster rate than List 2 and that, through a judicious assignment of programs to List 1 or List 2 according to their known characteristics, those programs most likely to keep the I/O channels busy will be given control more frequently. When List 1 degenerates to zero, the result will be straight rotational switching among the programs in List 2. This switching will be triggered by I/O interrupts and by voluntary releases of control. On the other hand, when List 2 degenerates to zero, straight rotational switching will occur among the programs of List 1, with the switching triggered only by voluntary release of control. It is therefore apparent that an imprudent assignment of programs to List 1 can defeat the purpose of concurrent operation because of the dependence in List 1 on voluntary release of control.

B. Program Release of Control

If a program requires the completion of a specific I/O request before it can continue operating, the following sequence is coded to relinquish control to EXEC:

```
TPO    f,packet address, Ø, $Tn:
LMJP   $B1, $WAIT                :
```

where f is the amount of film memory to save as follows:

f = 0, save 01 - 34₈, 101₈ - 117₈ and 130₈ - 177₈.

f = 1, save 01 - 34₈ and 101₈ - 117₈

f = 2, saving film is not required.

n is the type of packet as follows:

n = 1, I/O packet status testing

n = 2, communication packet status testing, and

\$WAIT1 is the entrance to the dispatcher for waiting for a specific request.

The instruction preceding the load modifier and jump instruction must be a test positive instruction and the h, i, and b designators must be zero.

Programs which use this release of control sequence are immediately returned to the active cycle upon completion of their specific request. If the interrupt which signifies completion occurs between the time of the TPO instruction and LMJP instruction, control will be returned immediately to the program.

If a program can operate after the completion of any one of a number of I/O requests, the sequence is:

```
TNG    Ø, packet address, Ø, $Tn:
CSJP   Ø, RESUME                :
TNG    f, packet address, Ø, $Tn:
CSJP   Ø, RESUME                :
LMJP   $B1, $WAIT                :
RESUME ...                      :
...    ...                      :
```

where f and n are as defined above, RESUME is the address to which control is to be returned and \$WAIT is the entrance to the dispatcher for waiting on any of a number of requests. The second instruction preceding the LMJP instruction must have an a designator of 0, 1, or 2.

Programs which use \$WAIT as the release entrance are returned to the active cycle immediately following the completion of any of their outstanding requests. If a packet was completed before a reference was made to \$WAIT, control is returned to the program immediately after the reference to \$WAIT.

Programs which attempt to release control to \$WAIT or \$WAIT1 with no outstanding request existing will be unconditionally terminated.

The program release termination codes are as follows:

- (16)g Illegal request (test instruction error)
- (17)g No outstanding request at time of release
- (76)g Job program has tested an illegal packet before releasing control to \$WAIT1

For program release termination conditions, the appropriate termination code and return address from B1 are stored in BØ film image for the terminated program.

C. The Switch Lists

An entry is maintained in either list 1 or list 2 for each program sharing the computer memory at any given time. The program is assigned to one of these lists, upon loading, according to its characteristics:

- 1) I/O limited programs to list 1
- 2) Compute limited programs to list 2

These characteristics are indicated in the PTY card of the job request. If the characteristics are not indicated, the program is automatically assigned to list 2.

The entries maintained for each program in the switch list contain the following information:

- 1) REENTRY ADDRESS: The address at which the program's operation was last interrupted and the one to which control is transferred when the program is again initiated.
- 2) FILM MEMORY INDICATOR: indicates to the dispatcher whether or not the contents of film memory must be restored for the program before it is given control. This indicator is set at the time of interruption of a program's operation, depending on the conditions under which the interruption occurs.

- 3) **MEMORY LOCKOUT INDICATOR:** specifies the quantity to be loaded into the memory lockout register before the program is given control to insure that all areas will be locked against attempted write-ins except those areas assigned to the program currently operating.
- 4) **WAIT INDICATORS:** these are set by the various sections of the Executive System to temporarily retire a program from the switching cycle. All wait indicators associated with a program entry must be cleared to zero before the dispatcher will again switch control to the program. Wait indicators are set for the following reasons:
 - a) In response to a type-in request to temporarily halt the execution of a program. (This indicator is cleared when a type-in is received requesting that program execution resume.)
 - b) As a result of a program voluntarily releasing control to \$WAIT1. (The indicator is cleared upon completion of a specified I/O or communications request.)
 - c) As a result of a program voluntarily releasing control to \$WAIT. (The indicator is cleared upon completion of any I/O or communications request.)
 - d) As a result of a RUSH job request. (The indicator is cleared after the RUSH job is completed and the program is returned to core memory.)
- 5) **OUTSTANDING REQUEST COUNT:** indicates the number of input/output and communications requests remaining to be processed.
- 6) **JOB REQUEST ID:** the Fieldata identification of the specified programs.
- 7) **ESTIMATED RUN TIME:** the estimated program run time in minutes. The operator is notified when this time is exceeded by the program.
- 8) **ERROR TABLE ADDRESS:** the beginning address of the program errors table.
- 9) **RUNNING TIME:** an accumulation of a program interrupt and compute time in milliseconds.
- 10) **PROGRAM TYPE INDICATOR:** indicates to the dispatcher whether the program is I/O or compute limited.
- 11) **TRACE MODE INDICATOR:** indicates to the dispatcher that trace mode must be reset prior to returning control to the job program.

D. Storing and Restoring Film Memory

Whenever the Executive System stores or restores film memory it is implied that the following is stored or restored:

Addresses (01)₈ through (34)₈, (101)₈ through (117)₈, and (130)₈ through (177)₈. Excluded from this range are the 16 input access control words and 16 output access control words [(40)₈ through (77)₈], eight unassigned registers [(120)₈ through (127)₈] reserved by EXEC to process the error interrupts, and three additional unassigned registers for EXEC usage [(35)₈ to (37)₈]. The real-time clock register address (100)₈ is available to programs for reading only.

Arithmetic carry and overflow conditions are tested and recorded when storing takes place. When the film memory is restored, carry and overflow conditions are reset according to the state they were in when the film memory was stored.

Due to the relatively large number of film memory registers which must be stored and restored when control is arbitrarily taken away from a program, considerable stress has been placed in the switching procedure on voluntary release of control with the program storing and restoring its own film memory registers.

IX. COMMUNICATIONS

The Communication section of the Executive System will handle all communication between the operator and the operating programs. This communication will take place via the computer keyboard and on-line typewriter on the console channel. Neither the keyboard nor the typewriter can be assigned to operating programs.

A. Communication Requests

To request a function of the Communication Section, the program must load a designated film memory register with a Request Parameter (described below) and execute a Load Modifier and Jump (LMJP) instruction to the Communication Section. The Request Parameter includes the address of the Execution Packet for the function to be performed. The Communication Section will store the request in the waiting list for the console channel and set the "status" in the Execution Packet to indicate that the request has not been completed. If there is no request currently in process, the requested function will be initiated. In either case, control will be returned to the requesting program at the address following the LMJP instruction.

The calling sequence used to enter the Communication Section is

8	9	FUNCTION	14	15	SUB FIELDS	37	
		L	D	P	\$Q α	:	
		L	M	J	P	\$B1, \$COM	:

where \$COM is the entrance to the Communication Section, and

α is the address of the Request Parameter.

B. Request Parameter

The Request Parameter specifies the address of the Execution Packet for the function to be performed and, if a chain of functions is being requested, the number of additional packets in the chain.

The Request Parameter may be written in the form:

8	9	FUNCTION	14	15	SUB FIELDS	37
		G			z/6, e/30	:

where e is the address of the first word of the Execution Packet, and

z is the number of additional packets in the chain, if any. (Maximum value is 6).

C. Execution Packet

The Execution Packet specifies to the Communication Section the function which is to be performed. Included within each packet is a 6-bit field which is used by the Communication Section to record specific codes to indicate to the requesting program the current status of each requested function. This code will indicate such conditions as "request in progress" or "request completed normally."

A Communication Execution Packet consists of a group of words arranged in a specific order according to the function which is to be performed. The words comprising the packet are of six basic types. Table 8 illustrates both the binary and symbolic forms of the Execution Packets corresponding to the six communications functions. An explanation of the six word-types follows:

Type 1: Function and Status Word

- f is a 6-bit code indicating the function to be performed. The function codes are as follows:

Octal

- 01 - Type
- 02 - Change
- 04 - Unload
- 10 - Load
- 20 - Type and Read
- 40 - Read

- s is a 6-bit code indicating the status of the request. The status codes are as follows:

Octal

- 00 - Normal completion ("Y" answer for load, unload, or change function)
- 10 - "N" answer for load, unload, or change function
- 11 - Exceed number of packets allowed for a chain
- 12 - Packet or buffer address outside of program limits
- 14 - Illegal channel or unit select
- 15 - Illegal function code
- 40 - Request has been listed
- 44 - Request requiring a reply is waiting for a reply

TABLE 8 COMMUNICATION EXECUTION PACKETS

FUNCTION	PACKET													
	WORD TYPE	BINARY FORM				SYMBOLIC FORM								
		35	30 29	24 23	18 17	12 11	0							
TYPE	1	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%; text-align: center;">f</td> <td style="width:25%;"></td> <td style="width:25%; text-align: center;">s</td> <td style="width:25%;"></td> </tr> </table>					f		s		F	f, Ø	:	
	f		s											
	2	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">r</td> <td style="width:25%; text-align: center;"> </td> <td style="width:25%; text-align: center;">n</td> </tr> </table>					00	r		n	H	r, n	:	
00	r		n											
3	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%;"></td> <td style="width:25%;"></td> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">d</td> </tr> </table>							00	d	H	Ø, d	:		
		00	d											
READ	4	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%; text-align: center;">f</td> <td style="width:25%;"></td> <td style="width:25%; text-align: center;">s</td> <td style="width:25%; text-align: center;">m</td> </tr> </table>					f		s	m	F	f, Ø	:	
	f		s	m										
	5	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">i</td> <td style="width:25%; text-align: center;"> </td> <td style="width:25%; text-align: center;">t</td> </tr> </table>					00	i		t	H	i, t	:	
00	i		t											
3	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%;"></td> <td style="width:25%;"></td> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">d</td> </tr> </table>							00	d	H	Ø, d	:		
		00	d											
TYPE AND READ	4	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%; text-align: center;">f</td> <td style="width:25%;"></td> <td style="width:25%; text-align: center;">s</td> <td style="width:25%; text-align: center;">m</td> </tr> </table>					f		s	m	F	f, Ø	:	
	f		s	m										
	2	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">r</td> <td style="width:25%; text-align: center;"> </td> <td style="width:25%; text-align: center;">n</td> </tr> </table>					00	r		n	H	r, n	:	
	00	r		n										
3	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%;"></td> <td style="width:25%;"></td> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">d</td> </tr> </table>							00	d	H	Ø, d	:		
		00	d											
5	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">i</td> <td style="width:25%;"></td> <td style="width:25%; text-align: center;">t</td> </tr> </table>					00	i		t	H	i, t	:		
00	i		t											
LOAD, UNLOAD, CHANGE	6	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%; text-align: center;">f</td> <td style="width:25%; text-align: center;">c</td> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">s</td> <td style="width:25%; text-align: center;">u</td> </tr> </table>					f	c	00	s	u	F	f, u	:
	f	c	00	s	u									
	2	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">r</td> <td style="width:25%; text-align: center;"> </td> <td style="width:25%; text-align: center;">n</td> </tr> </table>					00	r		n	H	r, n	:	
00	r		n											
3	<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:25%;"></td> <td style="width:25%;"></td> <td style="width:25%; text-align: center;">00</td> <td style="width:25%; text-align: center;">d</td> </tr> </table>							00	d	H	Ø, d	:		
		00	d											
		indicates that this portion of the word is used by the Executive for temporary storage.												

For communications error termination (status codes 11 through 15) the termination code is stored in the packet status indicator if the packet is within the program limits, and the appropriate code and packet address are stored in the BØ film image register of the terminated program.

Type 2: Output Control Word

r is the 16-bit address of the word containing the first character to be transferred as output. Bits 34 and 35 must be zero. Output characters are obtained from successive sixths of a word in order beginning with the most significant sixth. Succeeding words are obtained from consecutively increasing addresses.

n is the number of characters to be transferred as output. (The master space code (00) is not transferred as output. Although it is included in the total character count, it is ignored in the line count.)

Type 3: Chain Word

If a chain of packets is not desired, this word is optional with the exception of a TYPE AND READ packet, in which case it must be present but may be used as desired.

d is the 16-bit address of the first word of the next packet in the chain. Bits 16 and 17 must be zero.

Type 4: Function, Status, and Characters Accepted as input

m is the number of characters actually accepted as input.

f } are as described for the Type 1 word
s }

Type 5: Input Control Word

i is the 16-bit address of the word into which the first input character is to be transferred. Bits 34 and 35 must be zero. Input characters are stored in successive sixths of a word, in order, beginning with the most significant sixth. Succeeding words are stored at consecutively increasing addresses. If the last word of input characters does not contain 6 characters, the unused portion of the word is filled with Fielddata coded spaces.

t is the number of characters to be accepted as input.

Type 6: Function, Channel, Status and Unit

f } are as described for the Type 1 word
s }
c is a 4-bit absolute channel designator
u denotes a peripheral unit (by master bit selection)

D. Communication Functions

In addition to the more general communication functions which are to be described here, a standard set of messages and type-ins are available for communication between the operator and the Executive System. They are introduced where needed in other sections of this manual. All type-ins intended for the Executive System are prefixed by three letters which define the specific function.

The following general functions are available through the communication section of the Executive System:

1. TYPE: This function allows the program to request that a specified number of characters be transferred - as output to the typewriter from contiguous memory locations as specified in the execution packet. When the transfer has been completed, an indication will be recorded in the execution packet to reflect completion of the request.
2. TYPE AND READ: This function allows the program to request that a specified number of characters be transferred as output to the typewriter and that a specified number of characters be accepted as input from the keyboard in reply. The program's identity will be typed out preceding the program's message. The operator will be expected to type-in this identity preceding his reply. The type-in is deleted if the number of characters specified in the Execution Packet is exceeded. The message will be terminated by an end-of-message code on the keyboard.
3. READ: This function allows the program to request that a specified number of characters be accepted as input from the keyboard. The program's identity, message identity and the word "ACCEPT" will be typed out preceding the reply. When the operator's type-in has been completed, an indication of normal completion will be recorded in the request packet.

4. **LOAD:** This function provides the operating program with a standard method of notifying the operator that it requires a tape to be mounted on a specified tape unit, a paper tape to be placed in a paper tape reader, or a card deck to be placed in a card reader. The Execution Packet includes the absolute channel and unit numbers of the pertinent peripheral equipment together with the control word for typing out the label by which the tape or card deck may be identified. The Executive will notify the operator (via the typewriter) to load the specified device with the required tape or card decks. When the operator acknowledges that the loading has been accomplished, the Executive records an indication of normal completion in the Execution Packet.
5. **CHANGE:** This function provides the operating program with a standard method of notifying the operator that a magnetic tape is to be removed from a specified tape unit, and that a tape is to be mounted in its place. The Executive notifies the operator (via the typewriter) of the function to be performed, together with a type-out of the absolute channel and unit numbers of the specified unit. The tape label to be used is also typed-out.

When the operator acknowledges (via the keyboard) that the function has been accomplished, the Executive records an indication of normal completion in the Execution Packet.

6. **UNLOAD:** This function allows the program to notify the operator that it requires a tape to be removed from a specified tape unit, labeled in a certain way, and a master tape ring inserted if required. It is also used to notify the operator that a card deck is to be removed from a card punch and labeled, or that a paper tape is to be removed from a paper tape punch and labeled. The Executive notifies the operator of the function to be performed (via the typewriter) together with the channel number, unit number, and label to be used.

The functions, unload, load, and change require the following standard operator replies:

Y - indicates the requested operation has been completed

N - indicates the requested operation cannot be completed

E. Communication Conventions

The Executive System will type-out on the Monitor Printer whatever characters are typed-in on the Keyboard, on a character-by-character basis, i.e., as each character is received as input it will be sent as output to the Monitor Printer. This allows the operator character-by-character visual verification of each data character accepted as input, in order that he may immediately detect an error and take the necessary steps to correct it. The Executive System will not accept input from the Keyboard while the Monitor Printer is being used to output other data.

Output on the Monitor Printer requested through a communications Execution Packet will begin with a carriage return followed by three line feeds. This will be followed by six characters identifying the program from which the message originates. If the identity is less than six characters, it will be printed and followed by sufficient spaces to make the total six. For those messages requiring a reply, a space, followed by three alphanumeric characters assigned by the Executive to uniquely identify the message, will be typed. For those messages not requiring a reply, a space followed by three slashes will be typed. One additional space will then be typed. The time, requiring seven characters, and two spaces follow. These initial 20 characters of output are supplied by the Communication Section of the Executive System. The text supplied by the requesting program is then typed, beginning in the 21st character position.

The Executive will maintain a count of characters of text printed. When a carriage return is detected in the text or when the count of characters of text printed equals 50, the count is reset to zero, and a carriage return, a line feed and 20 space codes are sent to the printer. The typing of the text is then resumed, in the 21st character position. When the specified number of characters of text have been typed, the Executive types a special character to indicate the end of the message.

Each Keyboard input message will be initiated by the operator by depressing the INTERRUPT ENABLE button on the operator's panel and then the carriage return key on the Keyboard. If the Monitor Printer is not being used to output some other message, the Executive responds to the resultant External Interrupt by sending a carriage return code, two line feed codes and seven space codes to the Monitor Printer. The Executive then initiates input mode on the channel to allow the operator to proceed to type-in the desired input on the Keyboard. If the Monitor Printer is busy when the above mentioned External Interrupt occurs, the interrupt will be noted by the Executive and the response will be delayed until such time as the Printer is free.

As each input character is received it is typed-out on the Monitor Printer. The first three characters typed-in by the operator must be the alphanumeric identity of the output message to which he is replying or in the case of unsolicited type-ins to the Executive System, the three alphabetic characters identifying a standard type-in. After these characters have been typed on the Printer, the Executive automatically types one space, the time, and two additional spaces to align the following input text with the preceding input and/or output texts on the page. The operator may then proceed to type the input text which, in addition to being typed-out, will be transferred character-by-character to the core storage locations set aside for the input by the recipient program.

If at any time the operator wishes to cancel a partially completed input message, he may do so by depressing the INTERRUPT ENABLE button on the operator's panel and then a special "delete" code on the Keyboard. This causes the Executive to send the code to the Printer and to reset its tables to the state in which they were at the time the interrupt occurred, signalling the beginning of the message. Logically no input will have been transferred to the recipient program's input area, nor will the type-in have been initiated.

As in the case of output, the Executive maintains a count of the number of characters of input text typed-in. If a carriage return is typed-in or if the count equals 50, whichever occurs first, the Executive sends a carriage return, a line feed, and 20 space codes to the printer and resets the character count to zero. If the carriage return was typed-in, it will be stored in the program's input area as part of the input message. If the carriage return was generated by the Executive, the carriage return will not be stored as part of the input message.

The operator will conclude his type-in by depressing the INTERRUPT ENABLE button and then a special "end-of-message" code on the keyboard.

The characters□□□□□□ appearing on the monitor sheet indicate the operator has left the keyboard in the input mode for a period longer than twenty seconds without typing a character when a type-out is waiting. The type-in has been deleted and must be reinitiated. Other indications to the operator are as follows:

BUSY - indicates the unsolicited type-in must be re-initiated.

REPEAT - informs the operator that his last type-in was in error and he must reply again. The error may be a result of 1) an incorrect message identification response, 2) an invalid unsolicited type-in code, or 3) a response of other than Y or N to a LOAD, UNLOAD or CHANGE message. The console remains in the input mode under this condition.

Table 9 illustrates the formats of the type-outs and type-ins as they appear on the monitor printer. The symbols appearing in this table are:

iiiiii	the JOB REQUEST ID of the PTY Card
bbbbbb	the word "LOAD" or "UNLOAD" or "CHANGE"
tttttt	the time, in the range from 0000:00 to 2400:00
ddd	alphabetic symbol for unsolicited type-in
cc	channel number
uu	unit number
mm	message identification assigned by EXEC.

TABLE 9: MONITOR PRINTER MESSAGES

MESSAGE	EXPLANATION
TYPE-OUTS	
<p>iiiiiiΔPmmΔtttttttΔtext</p> <p>iiiiiiΔ//ΔtttttttΔtext</p> <p>ΔΔΔEXEΔXmmΔtttttttΔtext</p> <p>ΔΔΔEXEΔ//ΔtttttttΔtext</p> <p>iiiiiiΔPmmΔtttttttΔ"ACCEPT"</p> <p>iiiiiiΔPmmΔΔbbbbbbΔCccΔUuuΔtext</p> <p>ΔΔΔΔΔΔPmmΔtttttttΔtext</p>	<p>Program message requiring a reply.</p> <p>Program message not requiring a reply.</p> <p>Executive message requiring a reply.</p> <p>Executive message not requiring a reply.</p> <p>Program "ACCEPT" message requiring a reply.</p> <p>Load, Unload, or Change message.</p>
TYPE-INS	
<p>ΔΔΔΔΔΔPmmΔtttt.tttΔtext</p> <p>ΔΔΔΔΔΔXmmΔtttt.tttΔtext</p> <p>ΔΔΔΔΔΔdddΔtttt.tttΔtext</p>	<p>Reply to program message.</p> <p>Reply to Executive message.</p> <p>Unsolicited type-ins to EXEC.</p>

X. LOGGING

A logging medium may optionally be defined by the computer operator for the EXEC usage through the use of the LOG unsolicited message (see Table 11). If a logging medium is not defined, the operator is periodically reminded with the typeout:

```
LG UNDEF
```

If a logging medium is defined, the following four types of information can be recorded on it.

- 1) job initiation log
- 2) job termination log
- 3) trouble dump
- 4) I/O error log

This information except I/O error log is recorded in blocks of 257 words each and is available as input to a printing routine (EDIT) which will recognize the four formats and produce an edited High-Speed Printer output from them.

Whenever information is written on the logging medium, the EXEC checks to ensure availability of further space. When a warning point is reached, the computer operator will be informed by one of the following typeouts:

```
(number of words) CHG LG DRUM
```

```
(number of blocks) CHG LG TAPE
```

A new magnetic tape or drum area must be specified by the LOG message. If a new logging medium is not defined by the time the available space is used, the operator will be informed by one of the following typeouts:

```
INSUF RM LG DRUM *jrid*
```

```
INSUF RM LG TAPE *jrid*
```

The following general information applies to the four logging formats written on the log medium:

- 1) The first block of each format contains a distinct code in bits 35-30 on the first word. This code defines the data type in the block. Any subsequent blocks of the same data type will have 00 in bits 35-30 of the first word.

- 2) If an unrecoverable error occurs when attempting to write on the logging medium the operator will be so informed and the logging attempt aborted.

In case of error logging dump, the counters are not cleared.

- 3) Any "time" information is displayed in seconds.
- 4) All unit numbers (except for drum units) are in master bit format.
- 5) Any undefined area may contain garbage.

A. Running Time Log

The Executive System maintains a log of the internal processing time utilized by each operating program. The fact that I/O transfers have priority over operating instructions and variations in I/O loads will affect the number of instructions executed in a given real-time period causes the running time to be necessarily approximate. If the maximum operating time of the program is specified on the job request, the total running time of internal processing is periodically compared with the maximum operating time. If the time utilized exceeds that specified, the computer operator is notified by the typeout:

```
*jrid* EXCEEDED RN TIM
```

in the monitor printer. The operator can either halt or terminate the program by use of the HLT or TER messages (see Table 11) or the program can be allowed to continue.

The log of all running time is maintained in a table internal to the EXEC. Upon termination of a job, the total internal processing time will be displayed on the monitor printer. The estimated time from the job initiation log and the running time, as calculated by the EXEC, is included in the job termination log.

If at any time there are no programs which can operate until completion of an I/O function, the EXEC will record the unused internal processing time. The value of "wait time", accumulated from the time of loading EXEC, will appear in the job initiation log and job termination log. This information can be used to aid the scheduler in determining which programs run efficiently in parallel with each other.

B. Job Initiation Log

The job initiation log is taken at the completion of a successful load of a program. The format of the job initiation log is shown in Figure 10. The contents of the block are as follows:

- Word 0 - 41 in bits 35-30 indicates a job initiation log block.
- Word 1 - wall clock time in seconds (based on the time entered in the HRS message or based on 0000 if no HRS message was specified when the EXEC was initiated).
- Word 2 - job request identity (one to six Fielddata coded characters left-justified and space-filled).
- Word 3,4- program name (one to twelve Fielddata coded characters left-justified and space-filled).
- Word 5 - estimated run time in seconds as specified on the PTY card (in minutes) of the job request.
- Word 6 - name of program storage medium (one to six Fielddata coded characters left-justified and space-filled).
- Word 7 - location of program storage (either channel and drum address or channel and unit number); S1 contains an equipment code.
- Word 8 - bit 35 equals 1 for rush indication, bits 34-30 priority number, bits 23-18 sequence number, bits 29-24 precedence, and bit 0 equals one for compute limited and zero for I/O limited programs.
- Word 9 - bits 35-18 cumulative system time in seconds (since the EXEC was initiated), bits 17-00 cumulative wait time in seconds (since the EXEC was initiated).
- Word 10 - The first of from 1 to n data control words. Bits 17-00 contain the count of words of this data code type including the control word itself, and bits 23-18 contain a code identifying the data to follow.

0	41			
1	Wall Clock Time (sec.)			
2	Job Request Identity			
3	Program Name			
4	Program Name (Cont.)			
5	Estimated Run Time (sec.)			
6	Name of Program Storage			
7	E	C	Drum Address	
			Unit No.	
8	R	PTY	PRC	SEQ
				S
9	System Time		Wait Time	
10		DC	No. of words this code	

E = equipment code

Job Mix

(bits 23-18 of word 10 equals 01)

Job Request Identity	
Termination Indicator - not zero if job terminated.	Current Operating Time (sec.)

Facility Assignment

(bits 23-18 of word 10 equals 03)

Core		0 1	0 0 0 3
	IBANK Length	IBANK Address	
	DBANK Length	DBANK Address	

Other		0 3	No. of words this entry
	C		Unit No.

Drum		0 4	No. of words this entry
	C		Start Address
			Length

Jump Switches		0 2	No. of words this entry
	S1		Jump No.

- Code = 00 - no more data
- = 05 - the current data type is continued in the next block.
- = 06 - more data in the next block but it is of a new type.
- = 01 - two-word packets showing the jobs in the current mix in the first word of each entry and the amount of time charged to each job in H2 and a termination indication in H1 of the second word of the entry. H1 is not zero if the job program is terminating.
- = 02 - copy of the PARAM table of the job being initiated. From 1-10 groups of 11 words in Fielddata notation.
- = 03 - facility assignments of the job at time of initiation. Four classifications of facilities defined by a code in S4 (bits 17-11) of the first word of each.
 - S4 = 01 - core assignments
 - 02 - all jump switches assigned to program
 - S1 = 0 - switch not set
 - ≠ 0 - switch set
 - = 03 - equipment other than drum
 - = 04 - drum assignments.

If the write on the logging medium is unsuccessful, the following typeout will occur:

```
*jrid* INIT LG FAILED
```

C. Job Termination Log

A summary of information is written on the logging medium when a program is terminated regardless of the type of termination. The format of the Job Termination Log is shown in Figure 11. The contents of the block are as follows:

- Word 0 - 42 in bits 35-30 indicate a job termination block.
- Word 1 - Wall clock time in seconds
- Word 2 - Job request identity
- Words 3,4 - Program name
- Word 5 - The actual run time charged to the program (to the nearest second in binary notation)
- Word 8 - Termination or error code in H1
 - Program error termination - 775 octal
 - Operator termination - 776 octal
 - Normal termination - 777 octal
 - Exec Error termination - an appropriate code in H1

0	4 2	
1	Wall Clock Time (sec.)	
2	Job Request Identity	
3	Program Name	
4	Program Name (Cont.)	
5	Actual Run Time (sec.)	
6		
7		
8	Termination or Error Code	Pertinent Address
9	System Time	Wait Time
10	DC	No. of words this code

Figure 11. Job Termination Log Format

Word 9 - Pertinent address in H2
- cumulative system time in H1
- cumulative wait time in H2
Word 10 - same type of information as in Figure 10
and discussed in section B above for word
10 with the following exceptions:

- 1) No PARAM information
- 2) Jump switch settings under facility assignments will have a word for each of the 15 jump switches.

If the jump switch is not set, S1 = \emptyset
If the jump switch is set, S1 = the switch list number of the program to which it was assigned.

If the write on the logging medium is unsuccessful, the following typeout will occur:

```
*jrid* TERM LG FAILED
```

D. Input/Output Error Log

1. Error Logging Routine

A minimal system of logging of peripheral equipment errors will be maintained by the error logging routine. The errors will be kept by counts, in most cases, and by addresses in the case of parity errors on drum. A no response message will be sent to the operator whenever a count reaches its limit. Errors which are logged are for 1) each unit for drums and magnetic tapes, 2) each channel for drum, paper tape, card, and High-Speed Printer equipment, and 3) for each individual magnetic tape.

An error count table is maintained in core memory by the EXEC. The size of the table is one computer word for each channel except magnetic tape channels, two words for each magnetic tape unit, and an additional 21 words for each drum channel.

The following errors will be logged:

a. Magnetic tape errors (by individual tapes)

- 1) UNISERVO IIA tape unit two-block read (I/O error code 24)
- 2) UNISERVO IIIA tape unit improperly written block (I/O error code 54) (44 on IIIC)
- 3) parity errors (I/O error code 60) (44 on IIIC)
- 4) character count error (I/O error code 70)

When a tape is rewound, the error count will be recorded on the monitor printer if it has exceeded a predetermined value. The count of errors by tape unit will be incremented for error codes 24, 44, 54, 60 and 70 at rewind time if the limit is exceeded. Hence, the count by unit will show the number of tapes for a given unit which have exceeded the limit.

b. Magnetic tape errors (by tape units)

- 1) two-block read (I/O error code 24), write error (54 on IIIA, *45 on IIIC)
- 2) parity error (I/O error code 60) (44 on IIIC)
- 3) character count error (I/O error code 70)
- 4) channel synchronizer sequence error (I/O error code 20)
- 5) channel synchronizer character count error (I/O error code 30)
- 6) illegal function code (I/O error code 50) (34 on IIIC)

c. Magnetic drum errors

- 1) magnetic drum control unit or drum sequence error (I/O error code 60)
- 2) parity error during a continuous read (I/O error code 64)
- 3) magnetic drum control unit character count error (I/O error code 70)
- 4) end-of-block overflow word error (I/O error code 06)
- 5) parity error not during a continuous read (I/O error code 07)

For the 06, 07, and 64 drum I/O error codes, a table of addresses is kept by channel. These errors are grouped as parity errors and are not kept separately by code. Counts will be maintained for each address. The computer operator is notified on the monitor printer if the count reaches a predetermined value, or if the address table is in danger of overflowing. The I/O errors should be recorded on the logging medium at this time to prevent having the last address of the table overlaid by each succeeding new address.

d. Channel errors

- 1) channel synchronizer sequence error (I/O error code 20)
- 2) channel synchronizer character count error (I/O error code 30)
- 3) illegal function code (I/O error code 50)

*Although the interrupt code for write errors on the IIIC is 44, a "45" is used in the error logging dump to distinguish it from read parity errors.

2. I/O Error Logging Medium Format

The computer operator may cause a dump of the I/O errors onto the logging medium at any time through the use of the message:*

DER

The use of the message will clear the internal EXEC I/O error tables. The format of the I/O error log is shown in figure 12. The contents of the block are dependent upon the type of equipment for a given channel. Each channel which is in the equipment configuration (except monitor printer channels) will have an entry. The first channel entry is in word 2 and includes:

S1 = Equipment type code
= 01 - Magnetic tape IIA
= 03 - Magnetic tape IIIA
= 05 - High-Speed Printer
= 10 - Magnetic Drum
= 11 - Card equipment
= 15 - Paper Tape

Bits 29-26 = Channel number
S3 = one of four data codes as defined in figure 12.
H2 = count of the number of words in this channel entry including the channel word

The contents for each type of channel entry are:

a. Magnetic tape channels

Variable-size packets with unit number by master bit selection in bits 33-18 of word one. Words following contain hardware defined external interrupt (EI) code in S2 and a count (in binary) of the number of times this interrupt has occurred.

b. High-Speed Printer, paper tape and card channels

Variable-size packets which are similar to the magnetic tape packets except no unit number appears in word one of the packet.

* Another option is provided whereby the errors for one channel will be displayed on the console printer; log tables will not be cleared with the option:

DER . . . Ccc ⑤

c. Magnetic drum channels

Three types of packets are defined:

If S4 of the first word of the packet is:

- = 01 - The words following will contain an EI code in S2 and a count in T3.
- = 02 - The words following will contain a drum address at which a parity error occurred - in bits 22-0 and a count of the number of times - in bits 34-24.
- = 03 - The words following will contain an entry for each drum unit upon which a 60 or 70 EI occurred.

E. Trouble Dump

A trouble dump will occur if the EXEC terminates a program or if a program detects its own error and terminates by a load modifier and jump to \$ERR with a request for a dump (see section XII,B.).

The format of the logging medium for a trouble dump is shown in figure 13.

The contents of the trouble dump are:

1. First Block

- Word 0 - S1 = 40 octal indicates a trouble log
- Word 1 - Wall clock time
- Word 2 - Identity of job being dumped
- Words 3,4 - Program name
- Word 5 - Length and beginning address of IBANK assignment (binary)
- Word 6 - DBANK assignment
- Word 7 - S1 ≠ 0 - carry indicator was set
= 0 - carry indicator cleared
S2 ≠ 0 - overflow indicator was set
= 0 - overflow indicator cleared
- Word 8 - H1 - will contain a job termination or error code (See section X.C. job termination log)
H2 - an address which is pertinent to the error code.
- Word 10-54 - programs minor film (B, A, and R registers) - this will also be found in the dump itself but is reproduced here for ease of editing.

2. Subsequent Blocks

The blocks following will contain a binary dump of the assigned memory area of the program. If the write on the logging medium is unsuccessful, the following typeout will occur:

```
*jrid* TRBL DP FAILED
```

Ø	43				
1	Wall Clock Time (Sec.)				
2	E	C		DC	No. of Words this Channel

DC = Data Code

- = ØØ - End of Data
- = Ø1 - Channel Word
- = Ø5 - (Last Word in Blk) - Data For Channel Continued in Next Block
- = Ø6 - (Last Word in Blk) - New Channel Starts in Next Block

Magnetic Tape Channel

	Unit No. Master Bit	Ø1	No. of words this unit
	EI Code		Count

Drum Channel (count by channel)

		Ø1	No. of words this entry
	EI Code		Count

High-Speed Printer,
paper tape, card channels

		Ø1	No. of words this entry
	EI Code		Count

Drum Channel (parity error)

		Ø2	No. of words this entry
	Count		Drum Address

Drum Channel (count by unit)

		Ø3	No. of words this entry
	U		Count

U = Drum Unit No.

Figure 12. Error Log Dump Format

First Block

0	40	
1	Wall Clock Time (Sec.)	
2	Job Request Identity	
3	Program Name	
4	Program Name (Cont.)	
5	IBANK Length	IBANK Address
6	DBANK Length	DBANK Address
7	Carry IND	Overflow IND
8	Termination or Error Code	Pertinent Address
9		
10		
56	Contents of Program's Minor Film	
256		

Succeeding blocks contain binary dump of core areas assigned to program. First word of each block will contain 00 in S1.

Figure 13. Trouble Log Format

F. Date Blocks on Log Medium

If the operator has typed a date into EXEC via the DAT or HRS unsolicited message, the first block of the log information will contain the date information. The layout for this date block is as follows:

Word 0 - S1 = 44 octal

Word 1 - contains the month and day in Fielddata notation as given in \$FDAT (see section XIV)

Word 2 - contains the year in Fielddata notation as given in \$FDAT2 (see section XIV)

The remainder of the block is meaningless.

XI. DUMPING FUNCTION

The Executive System includes a facility for obtaining trouble dumps in case unexpected errors occur which cause premature termination of supposedly debugged programs. These dumps are provided only in connection with termination of the program run and are not intended as a substitute for dumps obtainable through normal debugging procedures. Trouble dumps are recorded on tape or drum for later printing on the High-Speed Printer. The format of the trouble dump is outlined in Section X.E.

A. Automatic Dump

An automatic dump is provided in conjunction with termination of a job if an error interrupt occurs which the job program is not prepared to handle or the Executive detects an error in a program. (See Section XII.)

The initial part of the dump includes the JOB REQUEST ID associated with the terminating program, the type of error detected, and the contents of the "P" register at the time of interrupt due to error or infinite loop. The remainder of the dump includes the contents of film memory, the state of the carry and overflow indicators, and the contents of all core locations assigned to the program.

B. Program Requested Dump

A trouble dump is also provided, if requested by a program in conjunction with abnormal termination of the job. In this case the dump includes the same information as automatic dumps with the exception that the "P" setting included is that which is recorded in index register B1 by the load modifier and jump instruction through which termination is requested. This address identifies the point in the program at which the termination with dump was requested.

XII. TERMINATION

A. Normal Job Termination

In order to terminate the operation of a program at the normal end of a job, control is returned to the Executive System through execution of the following instruction:

8	9	FUNCTION	14	15	SUB FIELDS	37	
		L	M	J	P	\$B1, \$END	:

When this reference is made, the program is removed from the switching cycle, entries pertinent to the program are deleted from the system tables, and all facilities assigned to the program are returned to available status, except those which have been transferred to another program.

The operator is notified of a normal job termination through a type-out on the Monitor Printer. The type-out includes the JOB REQUEST ID of the terminating job, the time of day, the internal compute time in military time, and the address recorded in index register B1 by the referencing instruction. The latter identifies the point in the program at which termination was requested.

B. Abnormal Job Termination

If a program is to be terminated for reasons other than the normal end of the job, e.g., because of the non-recoverable peripheral equipment error, the program may specify termination with or without a trouble dump.

The calling sequence for specifying such an abnormal termination is:

8	9	FUNCTION	14	15	SUB FIELDS	37	
		L	D	P	\$Q0, p	:	
		L	M	J	P	\$B1, \$ERR	:

where p is the address of a parameter word specifying the type of termination.

8	9	FUNCTION	14	15	SUB FIELDS	37
		G			d/1, 0-135	:

where d is either 1, specifying that a trouble dump is to be taken.

or 0, specifying that a trouble dump is not to be taken.

This type of termination is used in lieu of an "error stop" by programs operating under Executive control.

Termination procedures are similar to those described for normal job termination except a trouble dump of program onto the log tape may be requested. If the terminating program is part of a sequence then the job request is placed in suspension and no more jobs will be selected from this sequence until the operator exercises one of four options in regard to the suspended job request. See Table 10.

Abnormal job termination may also be initiated by the Executive in case an error interrupt occurs which a job program is not prepared to handle or the Executive detects an error in a program, e.g., an illegal parameter is submitted by a program. The former condition exists whenever the location in a program \$ERROR table corresponding to an interrupt which occurred, contains a zero address or some other address which is not legal for the program. In either case a trouble dump is automatically provided. If the job is in sequence then the job request and sequence are suspended.

C. Termination Specified by the Operator

The operator may specify termination of a job through a type-in on the computer keyboard. The format for the type-in is:

```
TER *jrid*
```

where TER identifies the type-in as a request for job termination, and "jrid" is the JOB REQUEST ID of the job to be terminated. The asterisk, which is optional, directs EXEC to supply a trouble dump on the defined logging medium.

D. Manual Termination of Program by Operator

If interrupts are disabled by a worker program and program retains control or if a worker program is in an endless chain of indirect references, the operator may terminate the worker program by executing the following procedure:

- 1) STOP computer
- 2) Save core address in P register
- 3) Press computer CLEAR button
- 4) Set P register to \$15
- 5) Press computer START. EXEC disables interrupts, locks in all of memory and stops with a load modifier and jump instruction waiting to be executed.
- 6) Clear P register and set to core address saved in step 2. Note, do not push COMPUTER CLEAR as this will clear the load modifier and jump instruction from function register.
- 7) Depress START button. The P setting is captured, and the program responsible for the trouble is terminated with a trouble dump on log tape. Dump includes film and P setting at time operator pressed STOP at step 1 above.

When a job program illegally jumps to an unallocated address one of the following things will happen:

- 1) It will sooner or later execute a store instruction and violate memory lockout, or
- 2) It will execute an illegal instruction, or
- 3) It will jump or return jump to \emptyset ,
- 4) It will jump to an interrupt location and simulate an interrupt.

See Appendix D for termination codes for cases 1 and 2 above. The terminations codes for cases 3 and 4 and manual operator termination are listed below.

Termination code	Meaning
00	Program has executed a jump or return jump to address \emptyset
77	Manual termination of program in instruction loop or in loop with interrupts disabled.
75	Program has jumped to an interrupt location.

E. Temporary Interruption of a Program

At any time the computer operator may temporarily halt the execution of a job program with the type-in:

HLT id

where the symbol HLT identifies the unsolicited type-in as a request to temporarily halt a job program, and id is the JOB REQUEST ID of the job to be halted.

Following this the computer operator has two options:

1. Resume execution of the program through the type-in:

PRO id

where id is the JOB REQUEST ID of the job in question and the symbol PRO identifies the type-in as a request to proceed with the execution of the job program. The PRO id type-in is also used to initiate the job program after loading is completed. (See Section V.A.4).

2. Terminate the program's operation.

F. Termination Typeouts

Upon occurrence of any of the four types of terminations defined in the preceding sections, EXEC produces a typeout on the console printer denoting the conditions associated with the termination. The format of this typeout is as follows:

$$\left. \begin{array}{l} \text{(NORM)} \\ \text{ERR} \\ \text{OPER} \\ \text{EXEC} \\ \text{SUS) SEQ} \end{array} \right\} \text{TER-*jrid*-address 1 Thhmm:ss} \left. \begin{array}{l} \\ \\ \\ \text{Eee address 2} \end{array} \right\}$$

where

jrid is the job request identity of the terminated job,

address 1 is either 1) for normal or error termination, the address following the load modifier and jump to \$END or \$ERR, 2) for operator termination or EXEC termination with a termination code of other than 01-07 or 51-57, the address of the next instruction to be executed if the program were to gain control, or 3) for EXEC termination with a termination code of 01-07 or 51-57, the last content of the B1 register of the job program.

hhmm:ss is the total time the job program ran.

ee is a termination code used for EXEC termination only. A complete description of the termination codes is given in Appendix D.

address 2 The address within the job program of either a packet in error or the next instruction to be executed in the next instruction plus one. For termination code 00 this address is probably meaningless.

The Eee address 2 fields are present in EXEC termination only.

The SUS SEQ field is present if the program is in a sequence and the termination is by other than a reference to \$END.

TABLE 10

This table shows the actions taken by EXEC for job requests, job programs and sequences of jobs in which errors are discovered. These errors and error message formats are listed elsewhere in this document. In addition, the status of facility transfers is shown and operator options for resolving the troubles are listed.

EXEC Action for Non-Seq.	Operator Options	Corresponding EXEC Action if Job in Sequence	External Transfers to Job	External Transfers from Job	Internal Transfers	Operator Options*
Job is suspended	SEL jrid DEL jrid	Job request and remaining sequence are suspended	Still intact in TRNT table	None	Transfers to job still intact	1. DELETE job request A. All facilities in state of transfer to job request are released B. Next job in sequence becomes a candidate for initiation
Job is terminated in error by operator, EXEC or job program	None - job and its request removed from system	Job program is removed from system but its job request is suspended thus serving to suspend sequence	None - if job is rerun, EXEC Loader will attempt to assign facilities received by transfer during initial run of job	Intact - will be released if same job is to be rerun. If job is deleted, then facilities will be transferred to next job in sequence	Transfers to and from job program may still be present; if program is rerun, facilities received during initial run may be transferred by operator	2. DELETE the sequence. All facilities in state of transfer to any job request of the sequence will be released 3. Select job with "SEL jrid" message. All facilities transferred from this job are released
Job is deleted (occurs when EXEC Loader cannot load job)	None - job request is removed from schedule	Load is terminated and job request is suspended; this also suspends sequence	Transfers intact	Intact - will be released if same job is to be rerun. If job is deleted, then facilities will be transferred to next job in sequence	Transfers to job are intact and there are no transfers from job	4. Replace suspended job request with new job request; see 3 above.

*When a sequence job request is suspended, the operator must exercise one of the four (4) options before more jobs are selected from the sequence concerned.

XIII. EXECUTIVE PROCEDURES

A. Initialization

The Executive System is stored on a system tape in ROC form. The Executive is initiated by mounting the system tape on tape unit zero and actuating the automatic bootstrap facility of the computer hardware. This causes the first block of the system tape to be read into consecutive core memory locations starting with address zero. The block thus loaded is the Executive Bootstrap routine. During the loading of EXEC, certain information is needed describing the environment in which EXEC is to operate. This information is needed to place the internal EXEC tables in their initial state. The parameters are defined in the beginning of the EXEC listing and must be entered by assembling EXEC as explained therein. In the event that incongruent information is entered, the message:

```
INITIAL ER xx REASSEMBLE EXEC
```

where xx is a code describing the type of error, is typed on the console printer and the computer executes an unconditional stop. This necessitates reassembly of the EXEC with a compatible configuration. If this message occurs during loading or operation of EXEC after the EXEC tape has been previously loaded with no errors the message is a result of a machine malfunction and should be ignored as far as reassembly is concerned. The Executive Bootstrap completes the loading of EXEC into core and drum and then notifies the operator that the EXEC is loaded with the message:

```
site name
```

When the Executive is initiated, it expects to receive from the operator initializing information such as date, time, and status of facilities. It also expects a specification of equipment which is not currently operable, and a specification of the facilities which are to be reserved for the Executive System.

The messages mentioned above are outlined in Table 11 and, except for the date and time messages, are further discussed elsewhere in this document. The format for entering the date and time is

```
DATΔmm/dd/yy  
HRSΔtttt
```

These two messages may optionally be combined into a single message in the following format.

HRS tttt Δ mm/dd/yy

where; mm is a one or two decimal digit specifying the current month.

dd is a one or two decimal digit specifying the current day of the month.

yy is the last two digits of the current year, and

tttt is the military time ($0000 \leq \text{tttt} \leq 2400$) expressed as a decimal number.

If the HRS message is not used to specify the time of day, EXEC considers the time as currently 0000.

Proper advantage of bootstrapping EXEC from drum storage is possible only if all non-EXEC systems do not destroy the first 130,000_g locations on the drum channel EXEC is stored on (this area is used for storing EXEC).

Non-EXEC systems, at the end of their run, may reinitiate EXEC in its initial state by executing the following sequence of coding:

```
IFNM d, a          d = drum channel of EXEC
FEXT d, Ø
FMJP d, $L
IMIN d, b
IFNM d, c
EAEI d, c
EIJP Ø, $L + 1
WAIT Ø, Ø
a   A   NI, 1, $L + 1
    F   TERM, Ø
b   A   I, $34Ø, Ø
c   A   NI, 1, $L + 1
    F   BOOT, Ø
```

The operator may reinstate EXEC via a manual bootstrap from the drum channel if the non-EXEC system does not use the above programmed bootstrap procedure.

There is one legal computer halt in EXEC. This halt is in the vicinity of 130100_g, or 170100_g for a 64K core, (the exact value is given by the P register). This halt occurs when the next EXEC overlay segment cannot be loaded from magnetic drum. To continue set Console Selective Jump 15 to reread the last overlay segment and depress computer START. The current function being attempted by EXEC will be aborted and EXEC will attempt to reload the control overlay segment from drum.

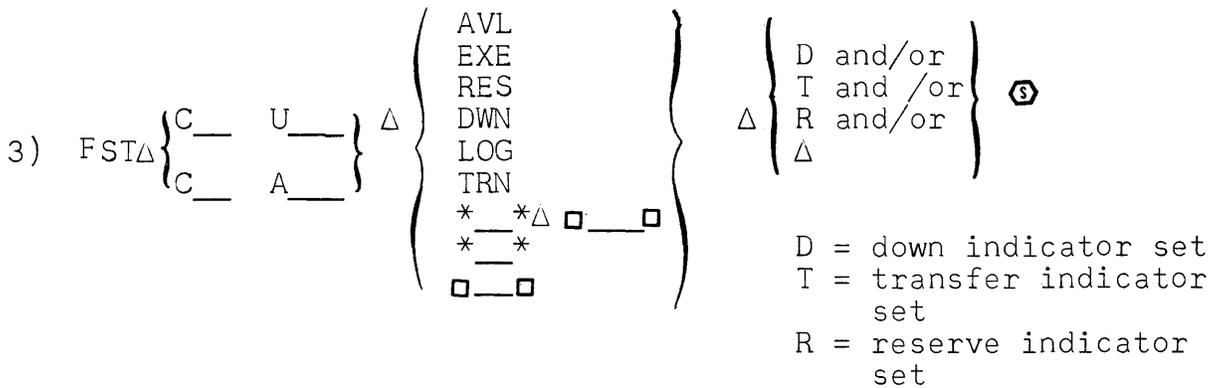
B. Unsolicited Messages

The message formats, usage and EXEC responses are shown in Table 11. The following is the definition of symbols used in the messages:

T	- tape
C	- channel
U	- unit
A	- address (drum)
D	- drum unit
L	- length (drum)
F	- final address (drum)
Δ	- indicates a space
R	- reserve number
* ___ *	- indicates job request ID bracketed by asterisks
□—□	- indicates program library name bracketed by squares.

The EXEC responses shown in Table 11 show only the variable part of the message. The total EXEC response will be in one of the following formats:

- 1) mc Δ ACK
where mc indicates the type-in function code (DWN, RES, REL, etc.)
- 2) mc Δ ee Δ IN Δ ER
where ee is one of the error code numbers shown in the table



- 4) mc Δ IN Δ ER Δ FST (appropriate variables from 3)

In addition to the error codes shown in Table 11 for each message type, the following error codes are common to several messages:

- 1 - field length is zero
- 2 - final drum address is present, but no initial drum address is given
- 3 - beginning and ending drum address produce a negative drum length
- 4 - drum length is entered twice
- 5 - unit number is too large
- 10 - a nondigit character is present in a numerical field.
- 12 - channel number is greater than 16.

C. Execution of Rush Jobs

The Executive system includes the capability to permit the running of a RUSH job as soon as is practically possible following the submission of the job request.

The following conventions are applicable to the use of the RUSH job request:

- 1) All input/output facilities required by the RUSH job request must be available.

TABLE 11. UNSOLICITED MESSAGES

Message format	Usage and definition	EXEC response
HRSΔtttt	Used to enter the time of day to the nearest minute. If no time is entered then the time is taken to start at zero.	ACK - normal 10 - nondigit character present in type-in
1. DAT mm/dd/yy 2. HRSΔttttΔmm/dd/yy	1. Used to enter the current date. 2. The second format is optional and if used, combines the HRS tttt and DAT mm/dd/yy messages.	mmmΔdd,Δ19yy - normal ex. MAR 7, 1963 7 - date contains a character other than digit or slash
1. SCHΔC ΔU 2. SCH C Δ U n2 n2 = a schedule identifier which must correspond to an identifier which appeared in columns 47-48 on the start card (valid only for magnetic tape)*	Used to define the location of the schedule input. Units which may be specified by this message are paper tape, magnetic tape and card readers. The job requests are read from the specified input device, checked for legality, and stored in the schedule of job requests. The existing schedule is updated to contain only uncompleted job requests. The SCH message may be used at any time.	*jrid*ΔSCHΔACK - normal FST - unit not available or reserved 16 - wrong equipment type specified 17 - start card in error 18 - console channel specified 20 - punch unit rather than read unit specified 31 - channel and/or unit number invalid 15 - possible machine error 22 - card count zero 23 - card count too large for specified drum area 21 - abnormal read (drum only) jrid of the first response is the job request ID of the last job request successfully read. If message is preceded by INT, the schedule was an internal transfer of a worker program.

*The complete field (columns 37-42) on the start card should be *REQnn, where nn = the alphanumeric identifier.

**If the schedule is on magnetic tape and an identifier was supplied with either the operator typein or in S1 and S2 of the parameter word of an internal transfer of the tape to EXEC for schedule purposes, the identifier is typed out with the error code, i.e., IDENT-nn.

TABLE 11. UNSOLICITED MESSAGE (cont.)

Message Format	Usage and definition	EXEC response
1. DWNΔC ΔU 2. DWNΔC ΔD 3. DWNΔC	Used to inform the EXEC that a unit is down. Down units will not be assigned to requesting programs. Units other than drum can also be given a down status by a D response to an I/O error message. A D response for a drum unit causes a 14 ₈ status code to be placed in the I/O packet but the facility table is not updated. Option 3 is used to down all units on a channel (valid for UNISERVO IIIA read-only channels).	response to type 1 ACK - normal 31 - channel and/or unit number invalid response to type 2 ACK - normal 15 - possible machine error 22 - channel number invalid 23 - drum number invalid 24 - no drum unit number typed 25 - no room in drum assignment table
1. UPFΔC ΔU 2. UPFΔC D 3. UPFΔC	Used to inform the EXEC that a previously down unit is up. This message has no meaning if the unit is not down and will result in an error message. If the unit is a magnetic tape unit and it was in an extended condition when it went down and it is not assigned to a job program, it will be rewound. Option 3 is used to declare all units on a channel as up (valid on UNISERVO IIIA read-only channels).	response to type 1 ACK - normal FST - assignment not available but unit is up; unless it goes down during rewind. 31 - channel and/or unit number invalid response to type 2 ACK - normal 20 - unit not down 22 - channel number invalid 23 - drum number invalid 24 - no drum unit number typed
1. EPLnIΔC ΔU n1 = library name 2. EPLnIΔC ΔU ΔP 3. EPLΔnIΔC ΔA ΔF/L ΔX 4. EPLΔnIΔC ΔL Δx 5. EPLΔnIΔC ΔR Δx x=A for UNISERVO IIIA =C for UNISERVO IIIC =I for UNISERVO IIIA	1) Used to inform the EXEC that a channel and unit contain a program library. The unit must previously have been in a reserved or available status. 2) The ending P defines a permanent library which can only be dropped on a DPL message. 3) The last three message formats are used to specify a drum area upon which a library tape with name n1 will be copied. The x field is needed if the library n1 is not currently part of the EXEC system.	response to type 1 and 2 ACK - normal IN ERROR FST - unit not reserved or available 20 - library already on drum 21 - channel does not contain magnetic tape or channel and/or unit number invalid 23 - library already assigned to different unit 22 - library already assigned to same unit 17 - library registration table presently filled.

TABLE 11. UNSOLICITED MESSAGE (cont.)

Message format	Usage and definition	EXEC response
	<p>The message format 5 is used to specify a previously reserved drum area (defined by length only) as the area to be used for copying the library tape.</p>	<p>response to type 3,4,5</p> <ul style="list-style-type: none"> ACK - normal 15 - possible machine error 16 - tape unit assigned to library is down, or attempt to re-wind the tape was not completed normally. 17 - library registration table is filled 20 - nl is already on drum, or a previous copy of this or another library is taking place. 21 - no core area presently available for copy routine or specified drum area is less than minimum (1000 words). 22 - specified channel is not a drum channel. 23 - invalid drum starting address or the library is presently in use. 24 - no drum area presently available which is large enough for library. 25 - drum allocation table filled. 26 - reserve number too large 27 - reserve number not defined 30 - drum area presently reserved (use message format 5) 31 - x field illegal or missing 32 - no tape unit of specified type presently available or reserved.

TABLE 11 UNSOLICITED MESSAGE (cont.)

Message format	Usage and definition	EXEC response
DPLΔn1 n1=library name	Used to drop a program library which was in use. The library tape will be rewound with interlock if the library is on magnetic tape. If the library is on drum, the area will be made available	ACK - normal if library is on drum FST - normal for other than drum 16 - n1 not in library registration table 17 - n1 not assigned to channel and unit or a drum area, cannot drop 32 - library presently in use, cannot drop 15 - possible machine error 20 - library has references but no assignment.
RPLΔn1Δn2ΔC__ΔU__ n1 = old library name n2 = new library name	used to replace a library on magnetic tape with a new library tape. The old library tape will be rewound with interlock. If the old program library was entered by option 2 of the EPL message, it was a permanent state and the new library will not assume this state.	ACK - normal IN ERROR FST - old library not assigned to channel and unit or channel and unit is down 16 - n1 not in library registration table 17 - n1 not assigned a channel and unit 20 - n1 on drum 32 - n1 in use 23 - n2 already assigned to another unit 24 - no room in library registration table 22 - n2 on drum
1. LOG C__U__ 2. LOG 3. LOG C__R__ 4. LOG C__A__F/L__ 5. LOG C__U__MD__ 6. LOG C__L__	Used to define the logging medium Message formats: 1) Will terminate old log tape if it exists; if none exists, this message defines the log tape. If old log medium was drum, it will be dumped on the tape defined and set the definition of the log medium to this tape. 2) Terminates a log tape; no new log medium defined. 3) Log medium established in a previously defined drum reserved area. 4) Log medium established on drum area specified.  the log was	ACK - normal IN ERROR FST - unit not available or reserved 15 - possible machine error 16 - waiting for rewind of old tape, resubmit message 17 - no LOG definition or on drum (2, 4, or 6) 20 - Unit specified down 21 - channel not assigned to magnetic tape 22 - channel and/or unit number invalid 23 - illegal starting address 24 - drum length specified too large for start  or length needed not available

TABLE 11. UNSOLICITED MESSAGE (cont.)

Message format	Usage and definition	EXEC response
	<p>previously defined on tape, the tape will be rewound.</p> <p>5) Log drum dumped on tape specified and log medium redefined to drum; tape specified will be rewound.</p> <p>6) Log medium established on drum according to the length specified. If the log was previously defined on tape, the tape will be rewound.</p>	<p>25 - no room presently available in drum table</p> <p>26 - reserved number currently not in use</p> <p>27 - reserved number improper; must be odd</p> <p>30 - area specified was previously defined by reserve length message; must use reserve option for message (C__R__)</p> <p>31 - abnormal read from drum, or abnormal write on tape.</p>
<p>1. RESΔC __ΔU __</p> <p>2. RESΔC __ΔL __</p> <p>3. RESΔC __ΔA __ΔF __</p> <p>4. RESΔC __ΔA __ΔL __</p> <p>5. RESΔC __</p>	<p>Used to reserve a unit or a drum area for future use. This message causes appropriate information to be placed in EXEC tables to prevent assignment to a working program. If the unit is already assigned, it will be reserved after it is released. Reserved units can be released only by an REL message.</p> <p>Drum areas may be reserved by length and a reserve number will be typed out. Any future references to this area must be by reserve number. Drum areas can be released by REL, LOG or TRN messages.</p> <p>Drum areas may also be reserved by address and length or final address</p> <p>Option 5 can be used to reserve all units on a channel (not valid for drum).</p>	<p>response to type 1</p> <p>ACK - normal</p> <p>FST - reserved unit is assigned</p> <p>31 - channel and/or unit number invalid</p> <p>response to type 2</p> <p>RES NO.__ - normal</p> <p>22 - channel number invalid</p> <p>26 - possible machine error</p> <p>27 - not enough room for length of reserve or all of drum is down</p> <p>30 - drum reserve table presently filled</p> <p>23,24,25 - possible machine error</p> <p>response to types 3 and 4</p> <p>ACK - normal</p> <p>IN ERROR FST - specified area not available</p> <p>22 - channel number invalid</p> <p>23 - address invalid</p> <p>24 - not enough room for reserved area</p> <p>25 - no room in drum assignment table</p> <p>15,26 - possible machine error</p>

TABLE 11. UNSOLICITED MESSAGES (cont.)

Message format	Usage and definition	EXEC response
<p>1. RELΔC__ΔU 2. RELΔC__ΔA__ΔL__ 3. RELΔC__ΔA__ΔF__ 4. RELΔC__ΔR__ 5. RELΔC__</p>	<p>Used to release drum areas and peripheral unit which were previously reserved. This message has no meaning unless it is preceded by an RES message.</p> <p>Drum areas which are reserved by length must be released by specifying the reserve number assigned by EXEC</p> <p>Option 5 can be used to release all units on a channel (not valid for drum)</p>	<p>response to type 1 ACK - normal FST - normal - assigned to something other than reserve</p> <p>31 - channel and/or unit number invalid</p> <p>response to types 2 and 3 ACK - normal IN ERROR FST - drum area specified not reserved</p> <p>22 - channel number invalid 24 - address invalid 25 - final address or length invalid</p> <p>response to type 4 ACK - normal 26 - reserve number invalid (too large) 27 - reserve number not presently in use, or not an odd number</p> <p>IN ERROR FST - possible machine error</p> <p>15, 22,24, 25 - possible machine error</p>
<p>1. TRNΔn1ΔC__ΔU__Δn2 2. TRNΔn1ΔC__ΔA__ΔF__ Δn2 3. TRNΔn1ΔC__ΔA__ΔL__ Δn2 4. TRNΔn1ΔC__ΔR__Δn2 5. TRNΔn1ΔC__ΔE__Δn2 n1 = Job request ID of receiving program n2 = parameter being transferred</p>	<p>Used to transfer a peripheral unit or drum area to a program which is currently running. The program must also request the unit or area through the normal transfer request system following the use of this message.</p>	<p>response to type 1 ACK - normal IN ERROR FST - unit not available or reserved</p> <p>16 - program not presently running</p> <p>31 - channel and/or unit number invalid 32 - transfer table presently filled</p> <p>response to type 2,3, and 5 ACK - normal IN ERROR FST - drum area not available or reserved</p>

TABLE 11 . UNSOLICITED MESSAGES (cont.)

Message format	Usage and definition	EXEC response
		<p>16 - program not presently running</p> <p>22 - channel number invalid</p> <p>23 - start address invalid</p> <p>32 - transfer table presently filled</p> <p>15 - possible machine error</p> <p>24 - no area on drum large enough for specified area</p> <p>30 - must use reserve option of message for drum area specified</p> <p>25 - no room in drum registration table</p> <p>response to type 4</p> <p>ACK - normal</p> <p>26 - reserve number too big</p> <p>27 - reserve number presently not in use</p> <p>16 - job presently not running</p> <p>IN ERROR FST - possible machine error</p> <p>30, 5, 15, 22, 23, 24, 32 - possible machine error</p>
<p>1. FSTΔC ΔU</p> <p>2. FSTΔC ΔA</p> <p>3. FSTΔC ΔR</p> <p>4. FSTΔC</p>	<p>1. Used by the operator to determine the status of a particular unit or drum address, or reserve number.</p> <p>2. Used by the EXEC to display the status of a facility in response to other unsolicited messages.</p> <p>3. Option 4 can be used to display the status of all units on a channel (also valid for UNISERVO IIIA read-only channels).</p>	<p>normal response for types 1, 2, and 3</p> <div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 10px;"> <p>C { U } Δ</p> <p> { A } Δ</p> <p> { R } Δ</p> </div> <div style="margin-right: 10px;"> <p>{ * } *</p> <p>{ * } * Δ □ □ □</p> <p>{ □ } EXE □ □</p> <p>{ □ } AVL</p> <p>{ □ } DWN</p> <p>{ □ } RES</p> <p>{ □ } LOG</p> <p>{ □ } TRN</p> </div> <div style="margin-left: 10px;"> <p>} D and/or</p> <p>} T and/or</p> <p>} R and/or</p> <p>} Δ</p> </div> </div> <p>error response to type 1</p> <p>31 - channel and/or unit number invalid</p> <p>15 - possible machine error</p> <p>error response to type 2</p> <p>15 - possible machine error</p> <p>22 - channel number invalid</p> <p>23 - address invalid</p>

TABLE 11. UNSOLICITED MESSAGES (cont.)

Message format	Usage and definition	EXEC response
		error response to type 3 26 - reserve number too large 27 - reserve number presently not in use 22, 23 - possible machine error
1. DER 2. DER C__	1. Dump of error log on logging medium; counts will be cleared. 2. The second format is used for displaying the error log for a specified channel on the console printer. The counts are not cleared for this case.	ACK - normal 15 - possible machine error 17 - log not defined 22 - too many errors for channel specified to dump on console; must dump on log 23 - not enough room on log tape 24 - not enough room on log drum 25 - abnormal write completion; error log not cleared
DEL * jrid	Used to delete a job or a sequence of jobs from the job request schedule. The * is used to delete a sequence of jobs.	ACK - normal 06 IN ERROR - jrid not in schedule RUN - the job to be deleted has been selected and may be running
HSL	Used to halt the job selection function of EXEC.	ACK - normal
SEL	Used to direct EXEC to begin selecting jobs again. This typein is the only way selection can be resumed after using the HSL message.	21 - Matched priority and precedence of an existing job, but new job has no sequence number. 22 - Matched priority and precedence of a nonsequence job.
HSL jrid	Used to suspend the identified job request; suspended jobs are not candidates for selection. if the job was in sequence the remainder of the sequence is also suspended.	ACK - normal 06 - jrid not in schedule 11 - jrid already selected 13 - jrid terminated or deleted

TABLE 11. UNSOLICITED MESSAGES (cont.)

Message format	Usage and definition	EXEC response
1. SEL jrid 2. SEL jrid lΔsppΔjrid2 jrid1 - job request identity of the permanent job. spp - sequence, priority, and precedence in this order. The alphabetic priority is the only part of the field which need be present. Specifying a priority of * causes the job to be run as a rush job jrid2 - job request identify under which the job is to be run.	1) Used to reinstate jobs that have been suspended and make them eligible as candidates for selection. If the reinstated job was in sequence, the remainder of the sequence is also reinstated. 2) the second format is used to select a job which was entered in the schedule as a permanent job.	ACK - normal 06 - jrid not in schedule 07 - spp field in fault 11 - jrid1 did not have permanent indicator set 14 - jrid not suspended 13 - jrid is deleted 16 - jrid2 field not specified 17 - jrid2 field is a duplicate to another jrid in the schedule 20 - no room in schedule table.
TPR	Used to terminate priority restrictions for selection of jobs. Use of this message causes program selection to be recycled.	ACK - normal
IPR	Used to reinstate priority restrictions for job selections. Use of this message causes program selection to be recycled.	ACK - normal
HLT jrid	Used to temporarily halt the execution of a job program.	06 - jrid not in mix
PRO jrid	Used to initiate execution of a program after loading is completed and to resume execution of a program temporarily halted by the HLT message.	06 - jrid not in mix
TER jrid TER * jrid	Used to specify termination of a job program. The optional asterisk directs EXEC to supply an informational dump on the logging medium.	program termination message 06 - jrid not in mix.

TABLE 11. UNSOLICITED MESSAGES (cont.)

Message format	Usage and definition	EXEC response
1. INF C 2. INF M 3. INF S 4. INF *S 5. INF L	1. First format is used to instruct EXEC to type all communications message numbers which are outstanding (need a reply). 2. Second format is used to instruct EXEC to type job request identity of each job currently in the mix. 3. Third and fourth formats are used to instruct EXEC to type the job request identities of all jobs in the schedule. The asterisk is used to include permanent jobs in the display. EXEC includes the following editing to the response to INFS or INF *S: a) a comma separates sequences and jrid's of jobs which are not in sequences. b) a plus sign separates jrid's of jobs in sequence which cannot be run in parallel. c) a colon separates jrid's of jobs in sequence which can be run in parallel. d) an asterisk precedes the jrid of a permanent job. e) a stop symbol (Ⓢ) precedes the jrid of a suspended job. f) a dollar sign precedes the jrid of a serial job. g) a vertical arrow (↑) precedes the jrid of a rush job. 4. Format 5 is used to display the names of all libraries currently registered with EXEC. The type of assignment (D for drum and T for Tape) and the number of jobs in the schedule referencing the library is also shown.	jrid's or message numbers None - no information to type in response to the query. 15 - possible machine error 16 - illegal option character.

- 2) Only one RUSH job request or sequence of RUSH job requests will be accepted at any one time.

Upon detection of a RUSH job request, the Executive will perform an I/O facility check. If the check fails, the operator is notified:

jrid FAILS RUSH

and the RUSH job will be retained as a candidate and given another facility check when any of the I/O facilities are made available. Following the above type-out, the operator may release reserved facilities needed by the RUSH job request. When the required facilities are available, core memory will be assigned to the RUSH job, which is then loaded and initiated in the normal manner.

Core memory is assigned to the RUSH job according to the following rules:

- 1) A normal core availability check is made. If the required core is available, then it is assigned to the RUSH job. Otherwise.
- 2) Determine the jobs that must be retired temporarily, to satisfy the core requirements of the RUSH job, and assign core memory normally.
- 3) Dump on drum or tape job programs identified in rule 2 above, and temporarily retire them from the switching cycle.
- 4) Load and initiate the RUSH job.
- 5) When the RUSH job terminates, the usurped core facilities and job programs are returned to their previous status, and entered into the switching cycle.

The above sequence of events is repeated for each RUSH job in a sequence of RUSH jobs.

D. Facility Transfer Function

The Executive System provides several methods for changing the assignment of facilities. These procedures are initiated by:

- 1) the job program
- 2) the computer operator
- 3) the job request TRN card

The ability to change the assignment of facilities permits sequences of functions to be performed. Some examples of these are:

- 1) Successive job programs may stack output on the same tape.
- 2) A basic set of data may be changed successively by several operating job programs.
- 3) Compiler and/or assemblers may transfer assembled jobs and job requests to the Executive for processing.
- 4) The computer operator may assign facilities to running programs for processing.

The facilities that may be transferred from one assignment to another are:

- drum areas
- magnetic tapes
- paper tape equipment
- High-Speed Printer
- card equipment

Core memory areas cannot be transferred; however, the contents of a core memory area may be transferred to a drum or tape facility, which may then be transferred.

The console channel cannot be assigned; hence it is not transferable.

The facility transfer function logically divides into two classes. They are:

- 1) External Transfer. This is handled exclusively through the use of "TRN" cards in the job request. The transfer is symbolic and accomplished by the EXEC Loader at load time. No cooperative coding in the object programs is required.

All external transfers are placed in a table internal to the EXEC. If the EXEC table will overflow, none of the external transfers from the job program just loaded will be entered in the EXEC table. The following message will occur to notify operator:

```
*jrid* TRN TBL OVFL0
```

- 2) Internal Transfers. These are not symbolic transfers and they are handled exclusively by cooperative coding between the two programs involved. The exception is that the operator via a type-in (Section D.2) may transfer a reserved or available facility to selected program.

1. Program Transfer

a. Initiation of Transfer

Operating job programs may transfer facilities to each other, to the Executive, and/or to successive jobs in the same sequence.

To request the transfer of a facility, the following sequence is coded:

8	9	FUNCTION	14	15	SUB FIELDS	37
		L, D, B			\$Q0, p, \$UOP	
		L, M, J, P			\$B1, \$REL	
		J, U, M, P			, C	

where p is the address of a Transfer Packet which details the transfer to be made. The Transfer Packet, illustrated in Figure 14, consists of four words as follows:

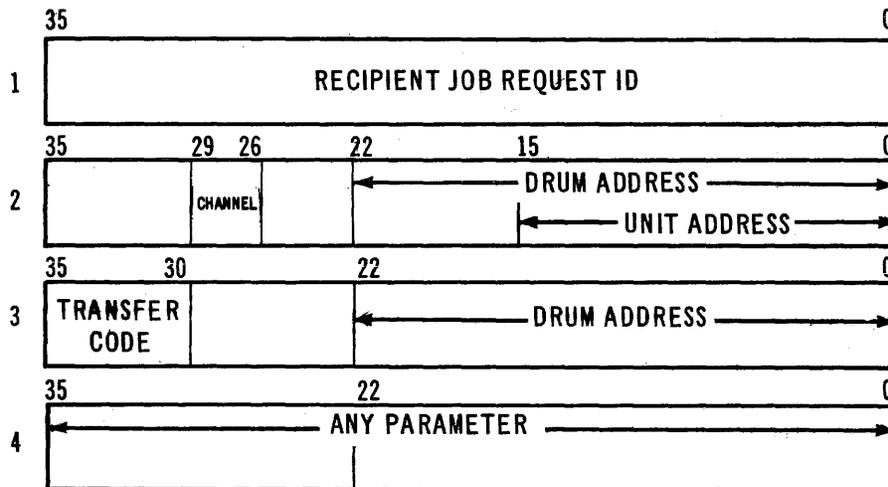


Figure 14: Transfer Packet

Word 1: Contains the job request ID of the job which is to receive the facility. The job request ID must be alphanumeric, left-justified, and space-filled on the right to a total of six characters. If facility contains a program library, word 1 contains the library name.

Word 2: Contains the channel and unit assignment for the facility or the channel and drum address if a drum facility is being transferred. The unit assignment is denoted by master bit selection.

Word 3: Contains a 6-bit transfer code which specifies the type of transfer to be made. The possible values of each of these six bits and their explanations are given in Table 12. Bits 22-0 contain the length of the drum facility.

Word 4: May contain any parameter agreed upon by job program using the facility transfer capability of the Executive. For transfer of a magnetic tape schedule, S1 and S2 of this word must be the alphanumeric identifier from columns 41 and 42 of the start card or if there is no identifier this word must be zero.

TABLE 12 TRANSFER CODE

BIT POSITIONS	VALUE	EXPLANATION
35	1	The facility is not a drum facility
	0	The facility is a drum facility
34	0	Null
	1	A program library is being transferred to EXEC.
33	0	Null
	1	Job requests are being transferred to EXEC
32	0	Null
	1	The facility is being released by the job program and, if tape, is rewound without interlock.
31	0	Null
	1	The facility is being transferred to a job program.
30	0	Null
	1	Rewind without interlock indicator for magnetic tape transfer.

The job request and program library indicators may be set concurrently. In this case, however, the job requests must be the initial data on tape, unless the fourth word contains a Fielddata name for the job request schedule. The name is two characters in bit positions 35-24.

- b. Completion of Transfer
Transfers to the Executive System are completed as follows:
- 1) Job Request transfers cause the Executive schedule function to process the requests from the specified facility and then make the facility available. Facilities allowable are magnetic drum and magnetic tape.
 - 2) Program library transfers are stored in the program library registration table.
 - 3) Facility release transfers cause the Executive to make the transferred facility available.

Transfers to other job programs are stored in the transfer table until the receiving program requests the next transfer. Two "transfer present" indicators are set in the receiving programs error table to indicate the presence of a transfer. These are \$ERROR+7, \$S2 and \$S3.

\$S2 - this indicator is set each time EXEC receives a facility transfer directed to the job program. It is cleared when the job program makes a \$TRN reference.

\$S3 - This indicator is set as long as EXEC is maintaining at least one facility transfer for the program.

If the receiving program is currently in a wait condition, it is placed on the active list eligible to receive control.

The job program can use any combination of the two options listed below, to obtain facilities being transferred to it.

The options are:

1. To release control at the \$TRN entrance, if no facility transfer is waiting.
2. To supply a transfer packet identifier to select the desired facility from the ones in state of transfer to requesting program.

The job program may use the indicators in ERROR+7 to determine when to reference \$TRN to receive a new facility. The calling sequence for \$TRN reference is:

1	TAG	7	8	9	FUNCTION	14	15	SUB FIELDS	37
					L D P			\$ Q Ø , r	:
	A				L M J P			\$ B 1 , \$ T R N	:
	A + 1				J U M P			, O U T	:
	A + 2								:
									:

The format of the word at r location is:

	29	24	17	00
	n	x	p	

where p - is the address of a three word buffer in the job program which receives the facility transfer packet if there is a find,

x - is reserved for determining the priority of \$XIO requests. (No conflict exists between \$TRN and \$XIO concerning the use of bits 29-18 of \$QØ. The programmer may place values permanently in \$S2 and \$S3 of \$QØ and subsequently load \$TRN or \$XIO packet addresses into the modifier portion of \$QØ). and

n = Ø - if no facility is found control is returned to A+1. If a facility is transferred to program, then control is returned to A+2.

n = 1 - If no facility is found, the program is placed in a \$WAIT condition and remains there till either 1) communication or I/O packet is completed and control returns to A+1 or 2) a facility transfer directed to the program is received by EXEC and is transferred to program and control is returned to A+2.

n = 2 - All facility transfers directed to this program are examined until a parameter match is made between the fourth word of the transfer packet and the parameter at p+2. If a match is made, the corresponding facility is transferred to the program and control is returned to A+2. If no match is found control is returned to A+1.

n = 3 - If a match is found the facility is assigned to the program and control is returned to A+2.

If no match is made the program is placed in a \$WAIT condition until

1) \$COM or \$XIO packet completion, then return to A+1.

2) A facility transfer with no parameter match is received by EXEC and control is returned to A+1.

3) Same as 2) above except a parameter match is made. The facility is assigned to the program and control is returned to A+2.

The TRANSFER CODE field of the completion packet is changed to one of the following octal values which define the type of facility:

VALUE	TYPE OF FACILITY
1	UNISERVO IIA tape unit
2	UNISERVO IIIC tape unit
3	UNISERVO IIIA tape unit
5	High-Speed Printer
10	Magnetic drum
11	Card Reader
12	Card Punch
15	Paper tape reader
16	Paper tape punch

c. Incomplete Transfer

When the recipient job program is deleted for any reason, all facilities in process of transfer to the deleted job are released to the available pool. The following notifies the operator:

```
DEL TRN TO *jrid*
```

```
Ccc△Uuu/Aaaaaaaaa△parameter
```

where jrid is the job request ID of the job which was to have received these facilities. The parameters are any that may have been agreed upon by the job programs.

When a suspended sequence job is selected by the operator, all facility transfers generated by a previous aborted load or run terminated in error are released. The message text is:

```
DEL TRN FROM *jrid*
```

```
Ccc Uuu/Aaaaaaaaa parameter
```

2. Operator Transfer

The operator may transfer reserved or available facilities to the Executive or to an operating job program. The types of transfers are:

- 1) A facility containing a set of one or more job requests may be transferred to the Executive.
- 2) A facility containing a program library may be transferred to the Executive System.
- 3) A facility may be assigned to an operating program. The Executive accepts transfer requests from the operator via the console keyboard. If the recipient job program is waiting for any packet completion, the job program is put into control and given an opportunity to reference \$TRN at a later point in time and pick up the facility. If the program is waiting for a facility transfer (Section D.1), the facility is assigned to the program, the 3-word transfer packet is placed into the reserved area at address p, and the program receives control at the second line following the \$TRN call. The format of the computer operators facility transfer request is shown in Table 11.

3. Job Request Transfer

This type of facility transfer is also restricted to jobs with equal priority and precedence values, and ordered sequence numbers. This function, however, depends upon data on the job request transfer card (TRN). The TRN card is necessary in the job request which uses facilities that are to be passed on to one or more succeeding jobs in a sequence or to the Executive System. Facilities containing job requests and/or a program library may be transferred to the Executive System through use of the TRN card. The TRN card is described in Section II.

4. Facility Transfer Errors

The error messages used for the transfers are listed following the typeout description and formats. The typeouts on the console printer contain pertinent data to be used in conjunction with the messages.

- 1) If the job program requested a facility transfer and before the transfer was completed it was erroneously assigned, then the typeout for message number one occurs.

TRN Ø1: *jrid* (assignment)

Program for job request (jrid) requested an internal facility transfer. The transfer packet was present, but the facility has been tagged with assignment (assignment) which replaced the previously assigned transfer status.

- 2) If none or more than one of the indicators for program library, job request file, facility release or transfer of facility to job request are set, then the typeout for message number two occurs.

TRN Ø2: *jrid* (channel) (unit or drum address)
(code)

The code is P for program library,
J for job request file,
F for facility release, and
T for transfer of facility to job request.

Program for job request (jrid) has released facility Cdd U/A dddddd and tagged it as (code) for Program library, Job request file, facility release or Transfer of facility to job request. The facility is being released.

- 3) If the facility cannot be found in the facility allocation tables, then the typeout for message number three occurs.

TRN Ø3: *jrid* (channel) (unit or drum address)

Program for job request (jrid) has referenced \$REL with the facility (Cdd) (U/A dddddd) which is not in the current configuration.

- 4) If the job program attempted to release a facility which did not belong to the job, the typeout for message number four occurs.

TRN Ø4: *jrid* (channel) (unit or drum address)
(assignment)

Program for job request (jrid) has illegally referenced \$REL with the facility (Cdd) (U/A dddddd) which has an assignment status of a job request ID or

6Ø = down
61 = transfer
62 = reserved
63 = library
64 = log medium
65 = job request file
ØØ = available

- 5) If the job program was releasing to the EXEC a program library that is already registered and assigned to a facility, the typeout for message number five occurs.

TRN Ø5: *jrid* (channel)(unit or address)(program-library name)

Program for job request (jrid) has released facility (Cdd) (A/U dddddd) to EXEC as(program library name). This library is already defined and assigned. Therefore, the additional program library is released, and if a magnetic tape, is being rewound with interlock.

- 6) If the recipient job request identity is not in the switch list table or schedule or if it is in the schedule but not in sequence or the same sequence as the releasing program, the typeout for message number six occurs.

TRN Ø6: *jrid₁* (channel)(unit or address)(jrid₂)
(error code)

Program for job request (jrid₁) has attempted to transfer facility (Cdd)(U/A dddddd) to job request (jrid₂). Error code is (error code). Facility is being released, and if magnetic tape, is being rewound with interlock. The error codes are defined as follows:

<u>Error Code</u>	<u>Meaning</u>
Ø1	Recipient job is not running or in the schedule.
Ø2	Recipient job is in schedule but not in same sequence of jobs (was an internal transfer).
Ø3	Recipient job is not in the schedule
Ø4	Recipient job is in schedule but not in same sequence of jobs (was an external transfer).

- 7) If the library registration table is full, the typeout for message number seven occurs.

TRN Ø7 *jrid₁* (channel)(unit or address)(jrid₂)

Library registration table is full, thus the program for job request (jrid) with facility (Cdd) (U/A dddddd) will not be registered. The recipient job request was (jrid₂).

- 8) If the recipient job has an incorrect assignment in the drum allocation table, the typeout for message number ten occurs.

TRN 1Ø: *jrid₁* (channel)(address)(jrid₂)(assignment)

Program for job request (jrid₁) has attempted to transfer facility (Cdd) (A dddddd) to job request (jrid₂), however, the facility had assignment (dd).

- 9) If an external transfer overrides an internal transfer, the typeout for message number eleven occurs.

TRN 11: *jrid₁* (channel)(unit or address)(jrid₂)

Program for job request (jrid₁) has attempted to transfer facility (Cdd) (U/A dddddd) to (jrid₂), however, an external transfer for this facility is overriding this internal transfer.

- 10) If the jrid of a transfer entry is not in the schedule or running, then the type-out for message number 12 occurs.

TRN 12: *jrid* (channel)(unit or drum address)
(assignment)

Program for job request (jrid) is not in the schedule and the facility (Cdd) (U/A dddddd) which has an assignment status of (assignment) is being released to available.

- 11) If the drum allocation table has been expanded to its limit, the typeout for message number 13 occurs.

TRN 13: *jrid₁* (channel)(address)(jrid₂)

Program for job request (jrid₁) has attempted to transfer facility (Cdd)(Addddd) to (jrid₂), however, the drum allocation table has reached its maximum expansion and can not contain an additional new entry.

- 12) If the transfer entry has a drum length specified which exceeds a maximum drum configuration, the typeout for message number 14 occurs.

TRN 14: *jrid₁* (channel)(address)(jrid₂)

Program for job request (jrid₁) has attempted to transfer or release the facility (Ccc)¹(Addddd) to job request (jrid₂), however, the length of the drum area is unreasonably large.

U77 indicates an improper unit designation.

E. Assembly (Compilation) and Testing of Programs

In the following paragraphs, the terms "assembly" and "assembler" should be considered synonymous with "compilation" and "compiler" respectively.

The assembly and/or testing of job programs can be handled in the manner described below. The philosophy behind this procedure is two fold. First a minimum of control interface between the assembler and Executive System is desired. This provides more flexibility to assemblers and a less complicated Executive System. Secondly, an environment for job program testing is provided which is almost identical to the operation of a proven program. The procedure is:

- 1) Job programs are prepared for assembly according to the conventions of the associated assembler. One job request to the Executive is necessary to load and operate the assembler.
- 2) The job request(s) for running the assembled programs, and the single object program output tape are transferred to the Executive System via one of the procedures defined in Section XIII.D.

- 3) All the object programs on the assembler output tape, for which there are job requests, will be run in the order specified by the priorities in their respective job requests. These job request priorities must differ only in their sequence numbers. See Section III.
- 4) Output tapes (or drum facilities) for stacking of debugging dumps from the jobs being tested can be transferred from job to job by use of one of the procedures outlined in Section XIII.D.

The testing of object programs that have been assembled at some previous time can be handled in a manner identical to the assembly and test procedure with the exception that a job request for the appropriate assembler need not be present.

F. Job Program Libraries

1. General

It is desirable and profitable to store many programs in ROC format on a single magnetic tape or consecutive group of magnetic drum registers. This greatly facilitates the loading of these programs for execution. The Executive System promotes this useful feature by allowing any named tape or drum facility to be defined as a facility containing a program library. Job requests for programs stored on these named program libraries may show this storage by placing the name of the program library in the MEDIUM NAME field of the PTY card. The Executive then loads the job program from the referenced program library. If the referenced program library has not yet been defined, then the Executive System will allocate a tape unit for this purpose and instruct the computer operator to place the named program library on the unit. This is achieved via a typeout:

```
LOAD Ccc Uuu WITH □library name□
```

The operator acknowledges this message with a

```
"Y" if the named program library has been loaded.
"N" if the named program library cannot be loaded.
```

If an "N" answer is received, the corresponding job request is suspended from the schedule.

The Executive maintains a count of all job request references to a named program library. This count is reduced by one each time a job program is loaded from the named program library. When the reference count reaches zero the facility (if magnetic tape) containing the library is rewound with interlock and released. The operator is notified with the message:

```
□library name□ Δ RELΔC__ΔU__
```

Program libraries stored on drum will not be released when their reference count reaches zero. Drum libraries may only be released by a "DPL" unsolicited message.

If the program library was defined as permanent by an EPL message, when the count of references reaches zero, the library is not released, nor rewound if on tape, and the following message is generated.

□library name□ ΔNΔREFΔCccΔUuu/Aaaaaaaa

This count is kept even though the program library has not been loaded. When the program library is loaded, its assignment will be stored in the previously established item containing the name and count of references.

2. Program Library Definition

Program libraries may be defined for the Executive System through the use of the internal and job request transfer functions (see Section XIII.D.) through a reference on the PTY card to a program library not yet defined and through computer operator type-ins.

The computer operator has three messages at his disposal to handle the assignment and the release of program libraries. These messages are:

- 1) Enter program library (EPL)
- 2) Drop program library (DPL)
- 3) Replace program library (RPL)

The format, usage and EXEC response for these messages are shown in Table 11.

3. Program Library Format on Magnetic Tape

Each job program must be in one of the formats acceptable by the 1107 Relative Load routine. The beginning of the tape may or may not contain an identification block. The Executive System does not check for an identification block. The program library tape is not restricted to storage of programs. Data may also be stored between the various job programs. Care must be exercised when data is placed on a program library tape. The first two words of a data block must not be equal to the 12 alphanumeric character identity of a job program or data library residing on the same tape.

A program library may be stored on magnetic drum in a variety of ways:

- 1) A tape-to-drum program
- 2) Output from assembler or compilers
- 3) Tape to drum copy through use of the EPL unsolicited message
- 4) Any other procedure which will produce a drum Program Library of the above format.

5. Program Library Housekeeping

All entries in the program library registration table are periodically checked for legality. All illegal entries are dropped from the table and the following error message is generated. The presence of one of these error message indicates possible machine error.

□ library name □ Δ REL Δ Ex

- | | |
|-------|---|
| x = 2 | library assigned to magnetic tape facility which is not in current configuration. |
| x = 3 | magnetic tape unit is not assigned to program library. |
| x = 4 | library assigned to drum channel not in current configuration. |
| x = 5 | drum address of library cannot be located. |
| x = 6 | drum facility not assigned to program library. |

6. Tape to Drum Program Library Copying

The tape to drum copy is initiated by use of one of the latter three EPL message formats illustrated in Table 11. The copying takes place in two parts.

The first part is the setup of the drum area, the assignment of a magnetic tape if the library name specified in the EPL message is not already registered, the selection of a 2K block of core in which the copy routine is to operate, and loading the copy routine.

The second part is running the actual copy routine. This routine may be anywhere in core and has a different switch list position than other portions of EXEC. It will run in parallel with other EXEC functions.

If a magnetic tape for the library was selected by EXEC, a load message will inform the operator which tape to mount and where to mount it. Responding to the load message with N restores all conditions to the original state and the copy will not take place.

EXEC copies only ROC programs to drum, everything else on the tape is ignored. In other words, the tape need not, but may be, in LIBRARIAN format. While the copy is taking place, a record of errors is maintained and they are all typed out (if any) when the end-of-file is reached.

The copy terminates with a response message which the operator must answer before the copy is completed. The message format is:

library name CPY-MORE?

If any errors have occurred from 1 to 9 error codes will be typed with this message. The error codes are given below.

The responses to this message are as follows (from 1 to 3 of these codes may be typed in at one time):

- Y - for yes there is another tape to be copied and added to this library on drum (should be typed in only after the new tape is mounted).
- L - to list the names on the console of the programs which have been copied. (If the name is a duplicate of a succeeding one it will be preceded by an asterisk.)
- X - cancel all copying which has been done to this point.
- N - No more copying is to be done. (NOTE - if a Y is typed it will cause the N response to be ignored.)

If the rewind of the tape which has just been copied is not completed normally or the drum area is filled, the Y response option is ignored.

The responses may be grouped in any order. Some examples of useful combinations are as follows:

1. XL - list of directory, new tape copied.
2. YLX - same as 1 except previous copy is cancelled.
3. XL - list of directory, cancellation of copy results so are.
The message will come out again allowing new options.
4. LN - list of directory, end copy.
5. LXN - list of directory, cancel copy, end any more copying.
6. YXN - cancel copy up to now, read new tape (N will be ignored).

The message is repeated until an N which is not ignored is typed in response.

Copy Error Codes

- 1 - the area on drum was filled. All the programs may not have been copied (Y option will be ignored).
- 2 - a) The limit of 85 programs was reached. All of the programs may not have been copied (Y option will be ignored).
b) There has been at least one duplication of program name - the second and succeeding programs of that name have been skipped.
- 3 - There was a checksum failure for one of the programs - program was skipped.
- 4 - At least one block for a program was not 256 words in length - program was skipped. In remote cases, it is possible for data (other than ROC programs) to produce this error.
- 5 - At least one block of a program did not read properly - program was skipped.
- 6 - The drum write of a program block was not completed normally - program was skipped.
- 7 - a) three abnormal reads in a row have taken place - copying for this tape is through.
b) If the write of the library directory was not completed normally (N response), the CPY-MORE? message will be typed out again with this error code added (try an N response again if the drum seems to be functioning properly or cancel with an X response).
- 8 - No ROC programs were copied.
- 9 - At least one block of a program was not of proper format - program was skipped.

G. Rerun Function

The Executive System provides a facility by which the operating programs may establish rerun points. The philosophy behind the rerun function provided by the Executive System is:

A rerun is primarily the responsibility of the job program.

A rerun point will be established in a manner and format which allows either the job program to restart itself when trouble is detected, or allows the Executive System to re-initiate the job program at an identified rerun point.

Responsibility for the actual rerun dump and for determining the status of all I/O facilities will remain with the job program.

The Executive will monitor the time of the rerun dump and supply pertinent information from the Executive tables.

1. Establishment of Rerun Point

The job program notifies the Execution of its intention to establish a rerun point by executing the sequence:

8	9	FUNCTION	14	15	SUB FIELDS	37	
		L	D	B	\$QØ, r,, \$UOP	:	
		L	M	J	P	\$B1, \$RRU	:

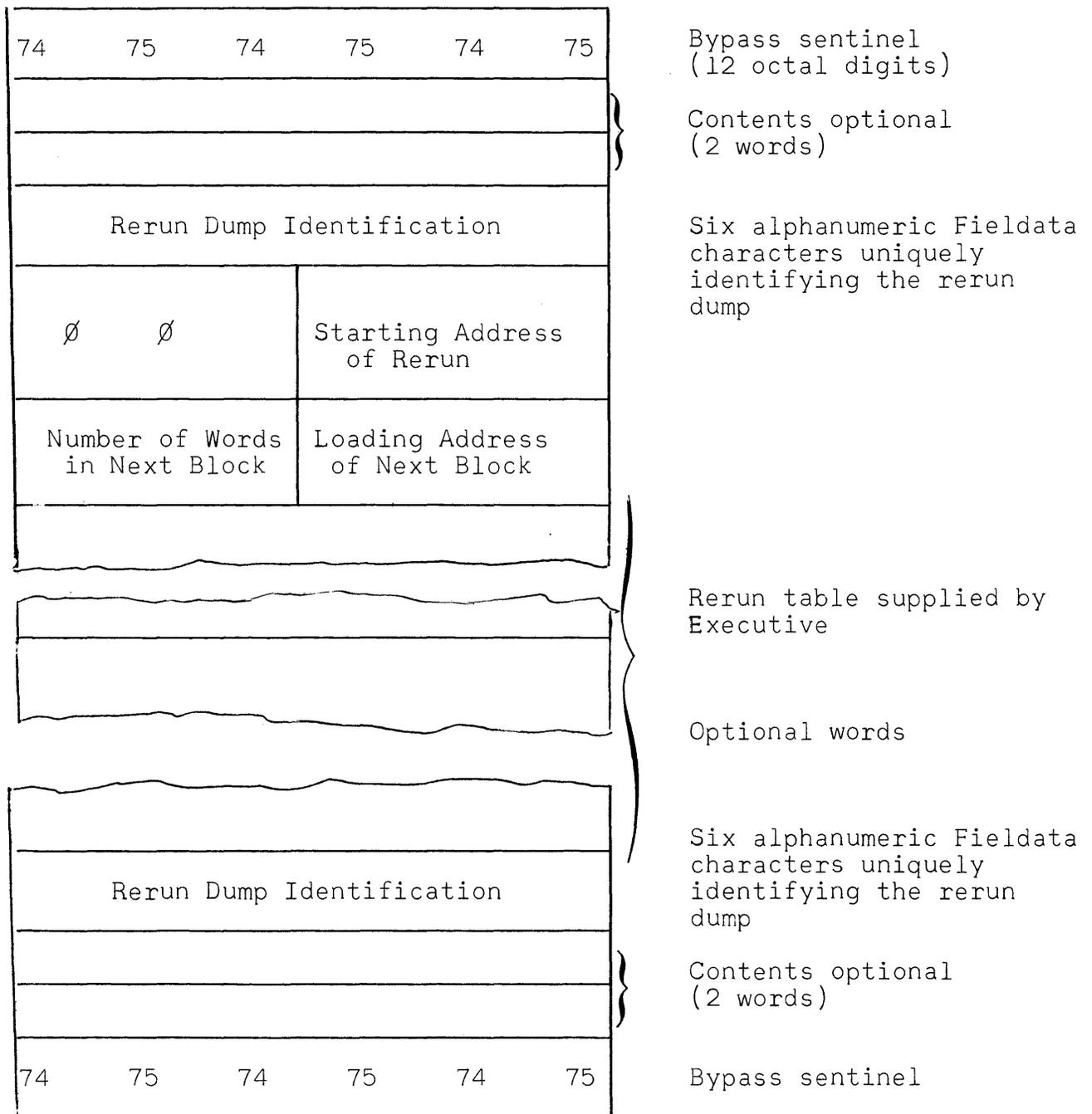
where r is the address in the program's assigned core area at which the Executive will store a rerun table containing information necessary to reset the Executive tables at the time of rerun.

The Executive checks for any outstanding input/output and communications requests from the requesting program. If there are no outstanding requests, the Executive System will generate a rerun table containing all data necessary to enable EXEC to reinitiate this program at some future time through the use of a job request. This table will be generated at the address specified in the QØ register. If any requests are outstanding, EXEC will wait until they are completed before generating the rerun table. After the table is generated, control will be returned to the job program.

If the rerun table will not fit in the core area assigned to the program, the job program will be terminated with an error code of 74₈ and an address of the location following the reference to \$RRU.

termination code
74₈

definition
The area assigned to the job program starting at the address in QØ is of insufficient length to contain the rerun table.



NOTE: The block length is variable to a maximum of 240 words

Figure 16. Rerun ID Block

- 1) Select a tape from its environment and write a rerun identification block, in the format shown in Figure 16.
 - 2) The second block must contain that portion of the job program that contains the rerun procedures for the job program.
 - 3) Remaining blocks are formatted according to the requirements of the job program. It is the responsibility of the job program to insure that all other blocks on the rerun tape are different from the rerun identification block in either the first or fourth words.
2. Initiation at Rerun Point: A job program can be reinitiated from a rerun point in two ways. The first is under control of the job program and the second is under initial control of the Executive and is completed by the job program. The format of the rerun dump is identical for both cases.
- a. Job Program Initiation: The job program locates the identification block, ignores the rerun table generated by the Executive, and reloads the remainder of the rerun dump according to procedures established by the job program itself.
 - b. Executive Initiation: The Executive System accepts Job Requests for programs that are to be initiated at an identified rerun point. The Job Request consists of the PTY Card in the form

```

JOB          RERUN          RERUN
REQUEST, PTY,, DUMP,, * , MEDIUM, E, PRIORITY.
ID          ID          NAME

```

where RERUN DUMP ID is the 6-character alphanumeric identity of the rerun dump.

RERUN MEDIUM NAME is the absolute location of the tape containing the rerun.

The procedure used by the Executive System to load and initiate a job at a rerun point is as follows:

- (1) The job is selected according to the rules of selection.
- (2) The operator is notified to load the rerun dump tape on a selected tape unit.
- (3) The bypass sentinel (Figure 16) is used as a search identifier to locate the rerun identification block. The RERUN DUMP ID field of the PTY Card is then compared with the fourth word of the block to verify that it is the rerun dump specified.
- (4) When the appropriate rerun identification block has been located, the data from the rerun table in the block is distributed to the proper Executive tables.

- (5) The next (second) block of the rerun dump is then loaded at the address specified in the sixth word of the rerun identification block.
- (6) Control is then transferred to the job program via a jump to the starting address specified in the fifth word of the identification block.

At this point the job program assumes full responsibility for completing the reload and continuing the run.

After the above procedure is completed the Executive returns to the normal concurrent mode of operation.

H. Independent Operation of Jobs

Occasionally special jobs may have to be run which can operate best only if they have full control of the computer. These jobs are handled in the following manner. A job request specifies an independent job to be run. The job to be loaded and run must be stored on drum or tape in absolute format, which can be loaded into core via a programmed bootstrap procedure. When this type of job is selected as the next candidate for initiation:

1. Job programs currently operating are allowed to terminate normally.
2. EXEC saves itself on its assigned drum channel. The first 50000 locations are used by EXEC. If these locations on drum are not damaged by the independent program, then EXEC may be reloaded via a programmer or manual bootstrap from the drum channel containing EXEC. If the log is being kept on drum, then the operator should define a log tape to which the drum log will be transferred.
3. A programmed bootstrap procedure is initiated on the specified channel.
4. When the independent job terminates, it returns control to the resident bootstrap routine which reloads the Executive System in its interrupted status and transfers control to it. If the terminating program destroys or fails to reference the bootstrap routine control is returned to the Executive by the operator via a manual bootstrap from the EXEC drum channel.

When EXEC is reloaded, the assumption is made that all equipment is still in the same status that it was prior to the independent run. The operator should inform EXEC of any changes. The execution of the job schedule will proceed normally and if the log was defined on tape prior to the independent run, EXEC will continue to log on that tape until notified otherwise.

5. A sample job request for an independent program is:

jrid, PTY,,,*, medium name, I, priority.

where, medium name is the absolute channel from which to bootstrap the independent program (format is Ccc).

XIV. DATE AND TIME

A. Date

The current date is available to worker programs in two different formats. Both formats are in Fielddata code.

1. Format 1

This is contained in one word tagged with the system tag \$DATE. The format is:

\$DATE =

mm	dd	yy
----	----	----

where,

mm = month. (1 = Jan, 2 = Feb, etc)
dd = two decimal digits for the day of the month, and
yy = two decimal digits for the last two digits of the year.

2. Format 2

This format is contained in two words. Each word must be accessed with a system tag and not a system tag plus increment.

\$FDAT =

l	l	l	Δ	d	d
---	---	---	---	---	---

\$FDAT2=

,	Δ	l	9	y	y
---	---	---	---	---	---

where,

lll = A three letter abbreviation for the month of the year. The abbreviations are:

JAN, FEB, MAR, APR, MAY, JUN,
JUL, AUG, SEP, OCT, NOV, DEC, and
dd and yy are as defined above.

\$DATE and \$FDAT will contain Fielddata space codes (05) if today's date has not yet been entered by the operator.

B. Time

The time of day in milliseconds may be obtained via the following calling sequence

LMJP \$B1, \$TIME:

Control is returned to the address of the calling sequence plus 1 and \$QØ will contain the current wall clock time. The exact time in seconds can be obtained by dividing QØ by 1000D.

APPENDIX A. IBM BCD Characters and Tape Codes (OCTAL)

CHARACTER	TAPE CODE	CHARACTER	TAPE CODE
A	61	V	25
B	62	W	26
C	63	X	27
D	64	Y	30
E	65	Z	31
F	66	∅	12
G	67	1	∅1
H	70	2	∅2
I	71	3	∅3
J	41	4	∅4
K	42	5	∅5
L	43	6	∅6
M	44	7	∅7
N	45	8	10
O	46	9	11
P	47	.	73
Q	50	-	40
R	51	\$	53
S	22	*	54
T	23	/	21
U	24	'	33

APPENDIX B. Job Request Error Codes (See Section II.C.)

An error in the job request format will cause one of the following messages to be typed on the console printer as explained in the text of this document.

SPΔ*jrid*ΔERnn or INTΔSPΔ*jrid*ΔERnn

The error codes, ERnn, and their description are as follows:

- ER1 Job request identity of current job too long. Message shows previous job request ID
- ER2 Card type field too long
- ER3 Illegal card type
- ER4 PM cards out of order
- ER5 TAL name too long
- ER6 TAL number too long or illegal character on the TAL card.
- ER7 Illegal character on TRN card
- ER8 TRN field too long
- ER9 FAC name too long
- ER10 Reserve on more than one card
- ER11 Illegal facility requested
- ER12 Illegal character on FAC
- ER13 Illegal sequence on FAC
- ER14 FAC asked for too large a core block
- ER15 Two FAC card FACILITY REQUIREMENT fields contain asterisks
- ER16 Job program name too long
- ER17 Illegal character or sequence on PTY
- ER18 Illegal number of characters in some field on PTY
- ER19 Rerun but no absolute units on PTY
- ER20 Card count on PTY and cards in job do not match
- ER21 ~~PTY cards out of grouping~~

APPENDIX B. (cont'd) Job Request Error Codes (See Section II.C.)

- ER22 FAC cards out of grouping
- ER23 TAL cards out of grouping
- ER24 TRN cards out of grouping
- ER25 First card of new jrid is not a PTY card
- ER26 Duplicate request ID
- ER27 No IBANK specified on FAC card
- ER28 Illegal RUSH field on the PTY card or FAC card is not the second card of the job request.
- ER29 Job request core storage table filled.
- ER30 PM number is not a numeric.
- ER31 Matched priority and precedence of a job in schedule but no sequence number on new job.
- ER32 Matched priority and precedence of a non-sequence job.
- ER33 Priority and precedence of Z63 but not a permanent job.
- ER34 FAC card ended with colon but next card is not FAC card starting with a tag field.

APPENDIX C. Loading Error Codes (See Section V.C.4.)

Error or warning conditions other than those connected with insufficient facility requests may be detected during the loading process. When an error or warning condition occurs, the operator is informed by a message on the console printer of the form

LD *jrid* ERR nnn or LD *jrid* WAR nnn

where nnn is a three digit error or warning code. The ERR message causes loading to terminate, but the WAR message does not. The error or warning codes are as follows:

- 00 Error in format or read of ROC program file
- 01 First block of program file not a label block
- 02 Checksum error on reading program file
- 03 Attempted to assign a segment storage drum table via a transfer or to transfer the assignment of a segment drum table to a succeeding program
- 04 Not EXEC ROC program
- 05 Table length modification not allowed
- 06 Jump switch cannot be deleted
- 07 I/O facility cannot be deleted
- 10 Illegal systems reference in main program
- 11 Nonrecoverable I/O condition
- 12 Machine or peripheral equipment error
- 13 Program file does not contain symbol of drum table being transferred to program
- 14 Program file does not contain symbol of drum table requested for transfer to a succeeding program
- 15 Program file does not contain symbol of I/O facility being transferred to program
- 16 Program file does not contain symbol of I/O facility requested for transfer to a succeeding program
- 17 I/O facility requested for transfer to a succeeding program was deleted
- 20 Table (DALL) containing drum table assignment full, cannot transfer drum table to succeeding program
- 21 Requested DBANK area too small
- 22 Requested IBANK area too small
- 23 \$PARAM table too small
- 24 Segment storage tape assigned to same unit as main or subroutine program file
- 25 No DBANK requested for load of a complex program
- 26 Subroutine requested segment storage
- 27 Tape from which subroutines are to be loaded is not a library tape
- 30 Library tape format error
- 31 Library directory too large for 2K buffer supplied by Loader
- 32 Subroutine or external reference not in library directory

APPENDIX C. (cont'd) Loading Error Codes (See Section V.C.4.)

- 33 Total drum area requested not enough to include segment storage
- 34 Attempted to transfer more than one I/O facility or drum table to same symbolic reference in program
- 35 External reference or subroutine not found in subroutine library
- 36 Subroutine medium not defined
- 37 Subroutine jump switch not defined in main program
- 40 Unable to assign subroutine I/O facility
- 41 Illegal systems reference in subroutine
- 42 Subroutine drum or core table not defined
- 43 Subroutine drum or core table longer than length of corresponding table of main program
- 44 Subroutine IBANK or DBANK longer than length given in INFO block of library
- 45 Warning, unable to make all table length changes specified on TAL card
- 46 Segment storage medium deleted
- 47 Attempted to transfer to an independent drum table, a drum area less than the area specified in a dependent table (a dependent table is one which has the same starting address as an independent one)
- 50 Core table length increment on TAL card too large (16-bit sum is less than original length)
- 51 Requested \$ERROR table length greater than length fixed by EXEC (97)

APPENDIX D. Termination Codes

Upon occurrence of the conditions listed below, the EXEC performs an error termination of the job program and provides a typeout to identify the packet of instruction address causing the error. The error codes listed below are included in the typeout.

The format of the EXEC termination typeout is as follows:

```
EXEC /// time EXEC TER *jrid*(address of next instruction to  
be executed) T (time) E (error code) (address in BØ film image)
```

1. Illegal Jumps and Interrupt Lockout

<u>Octal Code</u>	<u>Meaning</u>
00	Program has executed a jump or return jump to address Ø or 1
75	Program has jumped to an interrupt location
77	Manual termination of program in instruction loop or in loop with interrupts disabled

2. Internal Error Interrupt

<u>Octal Code</u>	<u>Meaning</u>
01	Illegal operation
02	Trace mode
03	Memory lockout (no recovery)
05	Characteristic underflow
06	Characteristic overflow
07	Divide overflow
51	Illegal operation and illegal recovery address in \$ERROR table
52	Trace mode and illegal recovery address in \$ERROR table
55	Characteristic underflow and illegal recovery address in \$ERROR table
56	Characteristic overflow and illegal recovery address in \$ERROR table
57	Divide overflow and illegal recovery address in \$ERROR table
61	Illegal trace mode set in area other than current program or EXEC entrance

For nonrecoverable error interrupt conditions, the appropriate code and P register setting are stored in the BØ film image for the terminated program.

3. Communications

<u>Octal Code</u>	<u>Meaning</u>
11	Exceeded number of packets allowed for chain
12	Packet or buffer address outside of program limits
14	Illegal unit or channel select
15	Illegal function code

For communications error conditions, the appropriate code and packet address are stored in the BØ film image for the terminated program.

4. Program Release

<u>Octal Code</u>	<u>Meaning</u>
16	Illegal release (test instruction error)
17	No outstanding requests at time of release
76	Job program has tested an illegal packet before releasing control to \$WAIT1

For program release error conditions, the return address from the B1 register and appropriate code are stored in the BØ film image for the terminated program.

5. I/O

The codes for I/O error termination are codes 20-37 in Appendix F.

6. Illegal Address

<u>Octal Code</u>	<u>Meaning</u>
74	One or more words of a transfer or release packet or rerun table addressed by the QØ register lies outside the core area assigned to the job program.

APPENDIX E. EXEC I/O Function List (See Section VII.)

Function	EXEC Mnemonic	Octal Value
UNISERVO IIA Tape Unit (variable block mode)		
Read forward	RTF	42
Read backward	RTB	62
Search forward	SRTF	46
Search backward	SRTB	66
Read forward with sentinel check	RTFS	43
Read backward with sentinel check	RTBS	63
Move forward	MTF	41
Move backward	MTB	61
Rewind	REW	20
Rewind with interlock	REWL	21
Write at 12.5 kc	WTL	01
Write at 25 kc	WTH	02
UNISERVO IIA Tape Unit (fixed block mode)		
Read forward	FRTF	52
Read backward	FRTB	72
Search forward	FSTF	56
Search backward	FSTB	76
Read forward with sentinel check	FRFS	53
Read backward with sentinel check	FRBS	73
Move forward	FMTF	51
Move backward	FMTB	71
Rewind	REW	20
Rewind with interlock	REWL	21
Write at low density	FWTL	03
Write at high density	FWTH	04
UNISERVO IIIA Tape Unit		
Read forward	RTF	42
Read backward	RTB	62
Search forward	SRTF	46
Search backward	SRTB	66
Read forward with sentinel check	RTFS	43
Read backward with sentinel check	RTBS	63
Move forward	MTF	41
Move backward	MTB	61
Rewind	REW	20
Rewind with interlock	REWL	21
Write	WTH	02
Masked search forward	MSF	47
Masked search backward	MSB	67
Contingency write	CW	03
Write end-of-file	WEFH	04

Function	EXEC Mnemonic	Octal Value
UNISERVO IIIC Tape Unit		
Write binary at high density	WBH	02
Write binary at low density	WBL	01
Write BCD at high density	WDH	06
Write BCD at low density	WDL	05
Write end-of-file at high density	WEFH	04
Write end-of-file at low density	WEFL	03
Skip while erasing	SKIP	10
Read binary at high density	RBH	42
Read binary at low density	RBL	37
Read BCD at high density	RDH	52
Read BCD at low density	RDL	47
Read binary high with sentinel check	RBHS	43
Read binary low with sentinel check	RBLS	40
Read BCD high with sentinel check	RDHS	53
Read BCD low with sentinel check	RDLS	50
Search binary at high density	SRBH	46
Search binary at low density	SRBL	44
Search BCD at high density	SRDH	56
Search BCD at low density	SRDL	54
Backspace block	BSB	61
Backspace file	BSF	64
Rewind	REW	20
Rewind with interlock	REWL	21
Magnetic Drum		
Read drum	RD	42
Block read drum	BRD	52
Search drum	SD	45
Block search drum	BSD	55
Search read drum	SRD	46
Block search read drum	BSRD	56
Chain block read drum	CBRD	62
Write drum	WD	02
Punched Cards		
Condition Fielddata input	CFDI	62
Condition column binary input	CCBI	63
Condition row binary input	CRBI	64
Read card	RC	41
Read card trip fill	RCTF	42
Read card trip fill with sentinel-check	RCTS	44
Condition Fielddata output	CFDO	04
Condition column binary output	CCBO	05
Condition row binary output	CRBO	06
Punch card stacker Ø	PCSØ	02
Punch card stacker 1	PCS1	03

Function	EXEC Mnemonic	Octal Value
High-Speed Printer		
Print HSP	PHSP	02
Print HSP variable format	PHSV	03
Paper Tape		
Read paper tape forward	RPT	42
Read paper tape backward	RPB	62
Punch paper tape	PPT	02
Control		
Remove logical interlock	RLI	07
Terminate	TERM	23

APPENDIX F. I/O Packet Status Codes (See Section VII.)

Before the Executive System returns control to a program which has executed the I/O request submission calling sequence, any former content of bits 35-30 of the first word of the I/O request packet is replaced by a new status code. The significances of the value of the codes used are listed below:

<u>Octal Code</u>	<u>Meaning</u>
00	The request has been completed normally. If there were any errors, recovery was successful.
01	Logical interlock was in effect at the time the request was to be executed.
02	A find was made in a read with sentinel check operation. Logical interlock has been set.
03	End-of-file (magnetic tape), invalid address during execution of function (drum), or an attempt was made to read a UNISERVO IIA tape backwards when it was in the rewind position. Logical interlock has been set for nondrum units. For a UNISERVO IIIC, a file separator was encountered during a forward read of search. If there is a block of data after the file separator, the tape is positioned ready to read that block. If the tape has not been recorded beyond the file separator, another forward read or search will result in a runaway tape. A backspace file operation will reposition the tape so that the tape mark is ahead of the read head, making it possible to back over the file with a backspace block operation. Status code 03 is also stored in response to backspace block if a tape mark is passed over.
04	End of magnetic tape, end of paper tape, or end of card deck. Logical interlock has been set.
05	End-of-block detected during a block search drum or block search read drum operation, i.e., a find was not made. For UNISERVO IIIC operation, an abnormal frame count error status code was received while reading the block just passed over; i.e., an exact multiple of six characters was not read. Due to the IBM recording formats, it may or may not indicate a read error. S6 of the last word read into core has a count of the number of significant characters in the last input word.

<u>Octal Code</u>	<u>Meaning</u>
06	A nonrecoverable error has occurred in reading the word following the drum end-of-block word. If cards were being read, an illegal character was found on the card most recently read. Logical interlock has been set in the latter case.
07	Nonrecoverable drum parity error. The address of the word in error is in bits 22-00 of the status word.
10	Not used.
11	A nonrecoverable read or write error has occurred, as specified by the interrupt code in bits 29-24 of the packet status word. Logical interlock is set for nondrum units. For block functions on magnetic tape, bits 35-18 of the record count word contain the number of blocks read or moved since the function was begun, including the erroneous block. The buffer status word reflects the number of words transferred into memory including those taken from the erroneous block.
12	Not used.
13	A read or write error on magnetic tape, card unit, or printer has resulted in an indeterminate position of the input or output medium. Logical interlock is set.
14	During the execution of the request, an interrupt indicating manual intervention was received. An EXEC I/O error message was typed out, to which the operator responded that the requested unit was down. Logical interlock is set for nondrum units.
15	The requested nondrum unit was declared down before the request was serviced.
16	While this request was in the request list, it was cancelled by a terminate request.
17	While the request was being executed, it was halted by a terminate request.
20	A priority specification other than 0, 1, 2, or 3 was in bits 35-18 of \$QØ.
21	The function code was not defined for the specified channel.

<u>Octal Code</u>	<u>Meaning</u>
22	The master bit unit designator field, 15-00 of word 2, contained no master bits, more than one master bit, specified a unit not part of the EXEC environment, or specified an input-only or output-only unit not consistent with the function code.
23	The specified unit was not assigned to the requesting program.
24	One or more words in the request packet, other than the first two, overlap a core area not assigned to the requesting program. The function code is used to determine the number of words which constitute a request packet.
25	Illegal input buffer starting address, or too many input words requested (cards).
26	The drum area to which writing is specified, includes locations assigned to another program.
27	An attempt was made to initiate a write on magnetic tape unit, with a zero word count, as specified in bits 33-18 of word 3.
30	Invalid starting address requested in drum read or search.
31	One or both of the first two words in the request packet overlap a core area not assigned to this program.
32	More than 65,535 output words requested in card punch operation.
33	Read requested from nonexisting drum address. (Operator response of F to I/O Error message 22.)
34	Write with BCD specified has been attempted on a UNISERVO IIIC with S1 of the first word equal to zero.
40	The execution packet address is stored in the request list.
41 - 77	The request is currently being serviced. If the execution packet contains fields indicating buffer status or number of blocks, lines, or cards processed, and if the requesting program has control, the requesting program may determine from these fields how many words of input or output have been processed so far.

APPENDIX G. Packet Formats and Calling Sequences

The following paragraphs outline the packets and corresponding calling sequences which serve as an interface between EXEC and the job program operating under control of EXEC. A complete description of the various operations can be found in the text of this document.

1. Input/Output Requests (See Section VII)

An I/O request is submitted to EXEC via the calling sequence:

8	9	FUNCTION 14	15	SUB FIELDS	37
		L D P		\$ Q 0 , a	:
		L M J P		\$ B 1 , \$ X 1 0	:
					:

where,

\$XIO is the entrance to the I/O section of the Executive system, and

a is the address of a request parameter.

The request parameter must contain the address of the request packet in H2 and the request list priority assignment in S3. T1 of this word is ignored.

The I/O request packet in generalized form is as follows. For specific usage, all of these words may not be needed. For instance, for a search on drum only words, 1, 2, and 6 are needed; hence the third word of the packet is coded as a sentinel word.

	35		30		26	24	22		18	15			
1	Status code	Interrupt code		Drum address								Status word	
2	Function code	Channel		Drum address								Function word	
				←Format→				←Unit number (master bit)→					
3	G	W						V					I/O Access word
4	G	W						V					Buffer Status word
5	Number successfully processed						Number to be processed					Record Count word	
6	Any identifier word											Sentinel word	
7	mask											Mask word	

2. Communications Request (See Section IX)

A communication request is submitted to EXEC via the calling sequence:

8	9	FUNCTION	14	15	SUB FIELDS		37
		L	D	P	\$Q0	, a	:
		L	M	J	P	\$B1, \$COM	:

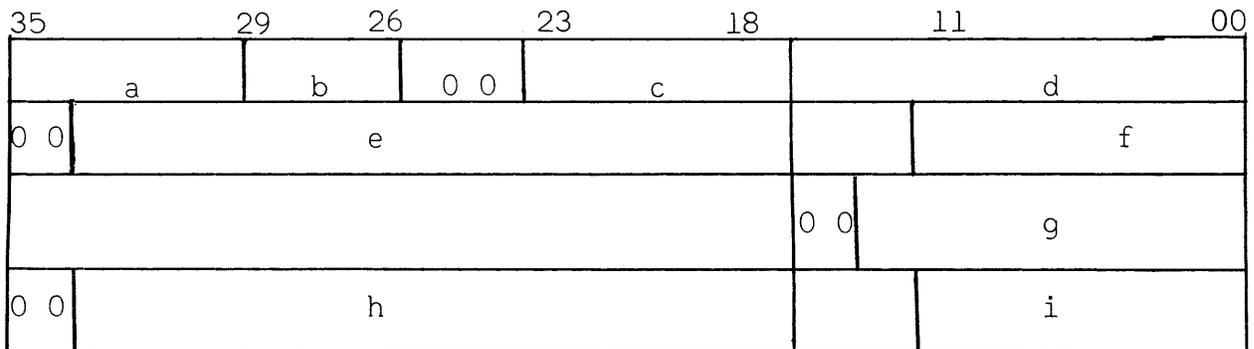
where,

\$COM is the entrance to the communication section, and

a is the address of the request parameter.

The request parameter specifies the address of the communication packet in H2 and the number of additional packets in the chain, if any, in S1. The remainder of the word, bits 29-18, is ignored.

The generalized form of the communication packet is as follows:



where,

a is the function code.

b is the channel number for the LOAD, UNLOAD, or CHANGE messages.

c is the status code.

d is the unit number, in master bit selection, for the LOAD, UNLOAD, or CHANGE messages.

e is the address of the first word of output (TYPE, TYPE AND READ, LOAD, UNLOAD, and CHANGE messages) or the address for deposit of the first word of input (READ).

f is the number of output characters to be typed or the number of input characters to accept.

g is the address of the first word of the next communication packet if the requests are being chained.

h is the address for deposit of input characters for the TYPE AND READ message, and

i is the number of input characters to be accepted as input for the TYPE AND READ message.

3. Internal Transfer of Facilities (See Section XIII.D.)

The internal transfer of facilities is performed by cooperative coding between the two programs involved. The donating program must release the facility and the receiving program must request the facility.

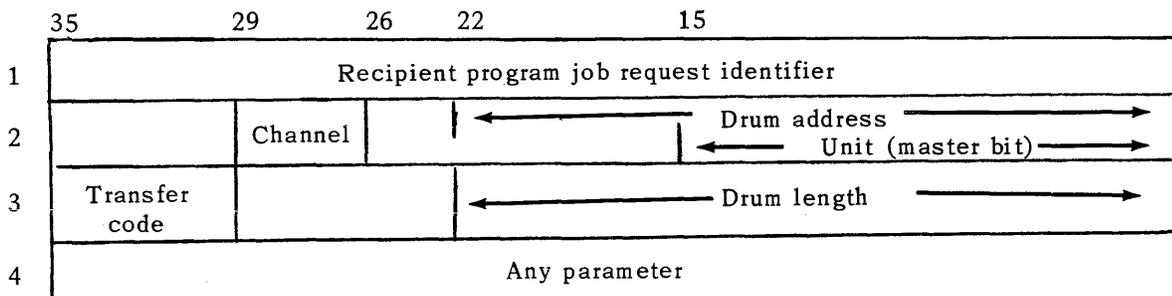
The donating program releases the facility by use of the following calling sequence:

8	9	FUNCTION 14	15	SUB FIELDS	37
		L D B		\$ Q Ø , p , , \$ U O P	:
		L M J P		\$ B 1 , \$ R E L	:
		J U M P		, C	:

where,

p is the address of the transfer packet.

The transfer packet format is as illustrated below:

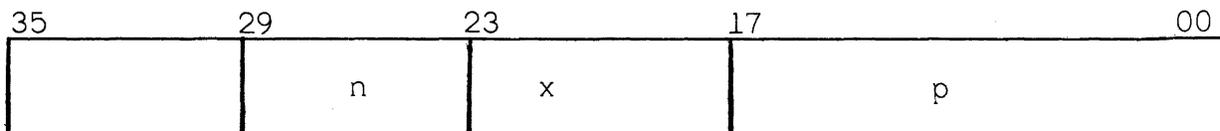


The EXEC maintains two indicators to show the presence of a transfer. These are S2 and S3 of \$ERROR+7 in the receiving program. S2 is set when EXEC receives a facility transfer directed to the job program and cleared when the transfer is completed. S3 is set as long as EXEC is maintaining at least one facility transfer for the program.

The job program may use the indicators to determine when to reference \$TRN to complete a transfer. The calling sequence to complete the facility transfer by the receiving program is:

1	TAG	8	9	FUNCTION 14	15	SUB FIELDS	37
				L D P		\$ Q Ø , r	:
	A			L M J P		\$ B 1 , \$ T R N	:
	A + 1			J U M P		, O U T	:
	A + 2						:
							:

where the word r must be of the form:



where,

p is the address of a three word buffer in the job program used for storing the received transfer packet.

x is used for determining the priority of I/O request, and

n = 0 if control is to return to A+1 if no facility is found or to A+2 if a facility is transferred.

= 1 if the program is to be placed in a wait condition until either 1) a communications I/O packet is completed at which time control goes to A+1 or 2) a facility transfer is completed at which time control is returned to A+2.

= 2 if all facility transfers directed to this program are examined until a parameter match is made between the fourth word of the transfer packet and the parameter at A+2. If a match is made the corresponding facility is transferred to the program and control is returned to A+2, otherwise control is returned to A+1.

= 3 If when a match is found the facility is assigned to the program and control is returned to A+2, and if no match is made the program is placed in a wait condition until either 1) a communication or I/O packet is completed at which time control returns to A+1, 2) a facility transfer with no parameter match is received by EXEC at which time control is returned to A+1, or 3) a facility transfer with a parameter match is made at which time the facility is assigned to the program and control is returned to A+2.

4. Program Release of Control Requests (See Section VIII.B.)

If a program requires the completion of a specific I/O request before it can continue operating, the following calling sequence is coded to relinquish control to EXEC:

8	9	FUNCTION	14	15	SUB FIELDS	37
T P O					f , packet address , Ø , \$ T n	:
L M J P					\$ B 1 , \$ W A I T 1	:

where,

f is the amount of film memory to save as follows:

f = 0, save 01-34_g, 101_g-117_g and 130_g-177_g.

= 1, save 01-34_g, and 101-117_g.

= 2, saving film is not required.

n is the type of packet being tested.

n = 1, I/O packet status testing.

= 2, communication packet status testing, and
\$WAIT1 is the entrance to the Dispatcher for waiting on a
specific request.

The instruction preceding the load modifier and jump instruction must be a test positive instruction and the h, i, and b designators must be zero.

Programs which use this release of control sequence are immediately returned to the active cycle upon completion of their specific request. If the interrupt which signifies completion occurs between the time of the TPO instruction, and the LMJP instruction, control will be returned immediately to the program.

If a program can operate after the completion of any one of a number of I/O requests, the calling sequence is:

8	9	FUNCTION	14	15	SUB FIELDS	37
		T	N	G	∅ , packet address , ∅ , \$ T n	:
		C	S	J	∅ , RESUME	:
		T	N	G	f , packet address , ∅ , \$ T n	:
		C	S	J	∅ , RESUME	:
		L	M	J	\$ B 1 , \$ WAIT	:

where f and n are as defined above, RESUME is the address to which control is to be returned and \$WAIT is the entrance to the Dispatcher for waiting on any of a number of requests. The second instruction preceding the LMJP instruction must have an a designator of either 0, 1, or 2.

Programs which use \$WAIT as the release entrance are returned to the active cycle immediately following the completion of any of the outstanding requests. If a packet was completed before a reference was made to \$WAIT, control is returned to the program immediately after the reference to \$WAIT.

Programs which attempt to release control to \$WAIT or \$WAIT1 with no outstanding request existing will be unconditionally terminated.

The program release termination codes are as follows:

- 16g Illegal request (test instruction error)
- 17g No outstanding request at time of release
- 76g Job program has tested an illegal packet before releasing control to \$WAIT1.

For program release termination conditions, the appropriate termination code and return address from B1 are stored in BØ film image for the terminated program.

5. Program Specified Termination Requests (See Section XII.)

In order to terminate the operation of a program at the normal end of a job, control is returned to EXEC through execution of the following job program specified calling sequence:

8	9	FUNCTION	14	15	SUB FIELDS	37
		L M J P			\$ B 1 , \$ E N D	:

When this reference is made, the program is removed from the switching cycle, entries pertinent to the program are deleted from the system tables and all facilities assigned to the program are returned to available status except those which have been transferred to another program. The operator is notified of a normal job termination through a type-out on the console printer.

If a program is to be terminated for reasons other than the normal end of the job, several options are available. The program may specify termination with or without a trouble memory dump and with or without deletion from the schedule of other jobs in sequence with the termination job. The calling sequence for specifying such an abnormal termination is:

8	9	FUNCTION	14	15	SUB FIELDS	37	
		L	D	P	\$ Q Ø , p	:	
		L	M	J	P	\$ B 1 , \$ E R R	:

where,

p is the address of a parameter word specifying the type of termination.

The word at p is of the form:

8	9	FUNCTION	14	15	SUB FIELDS	37
		G			d / 1 , Ø / 3 5	:

where,

d is either 1 specifying that a trouble memory dump is to be taken, or

0 specifying that a trouble memory dump is not to be taken.

If the terminating program is part of a sequence, then the job request is placed in suspension and no more jobs will be selected from this sequence until the operator exercises one of the four options in regards to the suspended job request. See Table 10.

6. Establishment of Rerun Point Request (See Section XIII.G.)

The job program notifies the Executive System of its intention to establish a rerun point by execution of the following calling sequence:

8	9	FUNCTION	14	15	SUB FIELDS	37	
		L	D	B	\$ Q Ø , r , \$ U O P	:	
		L	M	J	P	\$ B 1 , \$ R R U	:

where,

r is the address in the program assigned core area at which the EXEC will store a rerun table containing information necessary to reset the EXEC tables at the time of rerun.

The job program must provide for construction of the rerun identification block (Figure 16) and must provide for storage of the blocks of rerun dump on the external medium.

The job program can be reinitiated from a rerun point either under control of the job program or by submission of a job request to the EXEC.

APPENDIX H. Internal Transfer Codes (See Section XIII.D.)

Bit Position	Value	Explanation
35	0	The facility is a drum facility
	1	The facility is not a drum facility
34	0	Null
	1	A program library is being transferred to EXEC
33	0	Null
	1	Job requests are being transferred to EXEC
32	0	Null
	1	The facility is being released, and if tape, is rewound without interlock.
31	0	Null
	1	The facility is being transferred to a job program
30	0	Null
	1	Rewind indicator for magnetic tape transfer

The legal combinations for the transfer codes are listed below:

Octal Value	Control Bits					
	35	34	33	32	31	30
02	0	0	0	0	1	0
04	0	0	0	1	0	0
10	0	0	1	0	0	0
20	0	1	0	0	0	0
42	1	0	0	0	1	0
43	1	0	0	0	1	1
44	1	0	0	1	0	0
45	1	0	0	1	0	1
50	1	0	1	0	0	0
51	1	0	1	0	0	1
60	1	1	0	0	0	0
61	1	1	0	0	0	1
70	1	1	1	0	0	0
71	1	1	1	0	0	1

APPENDIX I. 1107 Subsystem Codes

Octal Code	Character printed		
	Console	Model 46 HSP	Model 751 HSP
00	ignored	nonprinting	@
01	~	np	[
02	%	np]
03	L.F.	np	#
04	C.R.	np	Δ
05	space	space	space
06	A	A	A
07	B	B	B
10	C	C	C
11	D	D	D
12	E	E	E
13	F	F	F
14	G	G	G
15	H	H	H
16	I	I	I
17	J	J	J
20	K	K	K
21	L	L	L
22	M	M	M
23	N	N	N
24	O	O	O
25	P	P	P
26	Q	Q	Q
27	R	R	R
30	S	S	S
31	T	T	T
32	U	U	U
33	V	V	V
34	W	W	W
35	X	X	X
36	Y	Y	Y
37	Z	Z	Z
40)))
41	-	-	-
42	+	+	+
43	<	<	<
44	=	=	=
45	>	>	>
46		&	&
47	\$	\$	\$
50	*	*	*
51	(((
52	"	np	%
53	:	:	:
54	?	np	?
55	!	np	!
56	;	'	;
57	Ⓢ	np	/

1107 Subsystem Codes (cont.)

Octal Code	Character printed		
	Console	Model 46 HSP	Model 751 HSP
60	0	0	0
61	1	1	1
62	2	2	2
63	3	3	3
64	4	4	4
65	5	5	5
66	6	6	6
67	7	7	7
70	8	8	8
71	9	9	9
72	'	'	'
73	;	np	;
74	/	/	/
75	.	.	.
76	□	np	□
77	↗	stop code	≠

APPENDIX J. EXEC I/O Request for UNISERVO IIIA Dual Channel Subsystem

The following rules are followed by EXEC I/O when servicing requests on UNISERVO IIIA dual channel subsystems:

1. Only the read-write channel can be specified in the I/O request packet. The read-only channel is used for servicing requests internally by EXEC and is not available directly to the job program.
2. When a request is submitted, the nature of the operation and the status of each of the channels is checked for appropriate action:

Type of Operation Requested	Channel, Status	Action Taken
Read	Read-only channel not busy	Service on read-only channel
Read	Read-only channel busy	Check read-write channel
Read	Read-write channel not busy	Service on read-write channel
Read	Read-write channel busy	Store request on list
Write	Read-write channel not busy	Service on read-write channel
Write	Read-write channel busy	Store request on list

3. When an interrupt signals the completion of a request on the read-only channel, the request list is scanned in search of a read request for a unit which is "ready". To be "ready", a unit must not be in operation on the read-write channel, nor may it be designated in a write request higher up on the list. If a read request for a ready unit is found, it is implemented on the read-only channel.
4. When the read-write channel completes a request, the request list is searched for a write request designating a unit which is ready. If the read-write channel cannot be made busy with a write request, another pass through the list is made in search of a read request designating a unit which is ready. (Note - All priorities are scanned for write requests before a read request is serviced on the read-write channel.)

The consequences of these rules are:

1. Requests for a given unit are always implemented in the order received, assuming that the priority specification (2, 1, or \emptyset) is not greater than that of a previous request still outstanding.
2. Write requests, when available, are given preference over read requests on the read-write channel.
3. When the request list contains only read requests, and the requests do not all designate the same unit, both channels are kept busy servicing read requests.

APPENDIX K
\$ERROR TABLE

<u>Decimal</u>	<u>Octal</u>	<u>Usage</u>
Ø	Ø	(unused error interrupt location)
1	1	illegal operation error recovery subroutine entrance
2	2	trace mode subroutine entrance
3	3	memory lockout error recovery subroutine entrance
4	4	(unused error interrupt location)
5	5	characteristic underflow error recovery subroutine entrance
6	6	characteristic overflow error recovery subroutine entrance
7	7	divide overflow error recovery subroutine entrance
8	1Ø	standard USE Subroutine Alarm Exit
9	11	IBANK length, IBANK, beginning address
1Ø	12	DBANK length, DBANK beginning address
11	13	carry designator
12	14	overflow designator
13	15	BØ
14	16	B1
15	17	B2
16	2Ø	B3
17	21	B4
18	22	B5
19	23	B6
2Ø	24	B7
21	25	B8
22	26	B9
23	27	B1Ø

\$ERROR TABLE (cont.)

<u>Decimal</u>	<u>Octal</u>	<u>Usage</u>
24	30	B11
25	31	B12-Q0-A0
26	32	B13-Q1-A1
27	33	B14-Q2-A2
28	34	B15-Q3-A3
29	35	A4
30	36	A5
31	37	A6
32	40	A7
33	41	A8
34	42	A9
35	43	A10
36	44	A11
37	45	A12
38	46	A13
39	47	A14
40	50	A15
41	51	contents of location 28_{10} or $34_8 = A15+1$
42	52	R1
43	53	R2
44	54	R3
45	55	R4
46	56	R5
47	57	R6
48	60	R7
49	61	R8
50	62	R9
51	63	R10

\$ERROR TABLE (cont.)

<u>Decimal</u>	<u>Octal</u>	<u>Usage</u>
52	64	R11
53	65	R12
54	66	R13
55	67	R14
56	70	R15
57	71	contents of location 88 ₁₀
96	140	contents of location 127 ₁₀

UNIVAC

DIVISION OF SPERRY RAND CORPORATION