UNIVAC 1100 Series

EXEC 8 Hardware/Software Summary

SPERRY UNIVAC COMPUTER SYSTEMS

UP-7824 Rev. 1

A

This document contains the latest information available at the time of publication. However, Sperry Univac reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Sperry Univac representative.

UNIVAC is a registered trademark of the Sperry Rand Corporation.

Other trademarks of the Sperry Rand Corporation include:

FASTRAND

MATED-FILM

PAGEWRITER

UNISCOPE

UNISERVO

 PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: UP-7824 Rev. 1

Section	Page Number	Update Level
Cover/ Disclaimer		Orig.
PSS	1	Orig.
Preface	1	Orig.
Contents	1	Orig.
1	1 thru 38	Orig.
2	1 thru 24	Orig.
3	1 thru 21	Orig.
4	1 thru 14	Orig.
5	1 thru 10	Orig.
6	1 thru 35	Orig.
Total	145 pages and covers	

Section	Page Number	Update Level

PREFACE

This booklet presents the most frequently used features of the UNIVAC 1100 Series Executive (EXEC 8) for the typical user programmer.

CONTENTS

PAGE STATUS SUMMARY
PREFACE
CONTENTS

1. CONTROL STATEMENTS 1–1
NOTATIONAL CONVENTIONS 1–1
SYSTEM DATA FILE (SDF) FORMAT 1–2
STANDARD PROCESSOR CALL STATEMENT 1–3
STANDARD PROCESSOR OPTIONS 1–3

2. EXECUTIVE REQUESTS (ERs) 2–1
ER FUNCTIONS AND FORMATS 2–1

3. LIBRARY SUBROUTINE PACKAGES 3–1
BLOCK BUFFERING PACKAGE (BBP) 3–1
DYNAMIC DUMPS 3–5
PROGRAM TRACE ROUTINE (SNOOPY) 3–10
EDITING PACKAGE FOR IMAGE COMPOSITION (EDIT$) 3–13
ASCII IMAGE COMPOSITION EDITING PACKAGE (AEDIT$) 3–19

4. GENERAL ERROR AND STATUS INFORMATION 4–1
ERR MODE (EMODE) AND I/O STATUS CODES 4–1
CONTROL STREAM DIAGNOSTIC MESSAGES 4–6
FACILITY REQUEST STATUS CODES 4–12

5. DEMAND PROCESSING 5–1
GENERAL DEMAND TERMINAL OPERATIONAL PROCEDURES 5–1
TELETYPEWRITER/DCT 500 SYMBIONT 5–2
DCT 500 IN TELETYPEWRITER MODE 5–4
UNISCOPE 100/DCT 1000 SYMBIONT 5–5
UNISCOPE 300 SYMBIONT 5–7
TERMINAL SECURITY SYSTEM 5–8

6. HARDWARE CHARACTERISTICS 6–1
ASCII AND FIELDATA CONVERSION TABLE 6–1
INSTRUCTION REPERTOIRE 6–4
MNEMONIC/FUNCTION CODE CROSS-REFERENCE 6–15
OCTAL VS. MNEMONIC INSTRUCTION CODES 6–16
MNEMONIC DESIGNATION AND ABSOLUTE ADDRESSES OF CONTROL REGISTERS AND PARTIAL WORD DESIGNATORS 6–17
CONTROL REGISTER ADDRESS ASSIGNMENTS 6–18
GENERAL REGISTER STACK (GRS) 6–20
j-DETERMINED PARTIAL WORD TRANSFERS 6–21
PROCESSOR STATE REGISTER (PSR) FORMAT 6–24
PROCESSOR STATE REGISTER EXTENDED (PSRE) FORMAT 6–25
OTHER WORD FORMATS 6–25
FIXED ADDRESS ASSIGNMENTS 6–29
HARDWARE STATUS CODES 6–31
UNISERVO 12/16/20 SENSE DATA BYTES 6–31
DISC HARDWARE STATUS CODES 6–32
MSA STATUS WORD FORMAT 6–33
MSA AUXILIARY STATUS WORD FORMAT 6–34
POSITIVE AND NEGATIVE POWERS OF 2 (BINARY), 8 (OCTAL), and 16 (HEXADECIMAL) 6–34

1. CONTROL STATEMENTS

NOTATIONAL CONVENTIONS

The general EXEC 8 control statement format is as follows:

Brackets are used to indicate optional fields or subfields.

The operation field is terminated by one or more spaces.

The operand fields specify parameters associated with the command fields. These are separated by commas and are specified by the user as dictated by his requirements. The content of each operand field, the number of operand fields, and whether each is required or optional varies with the command selected. Operand fields, in turn, may contain parameter subfields that are separated by a slash (/). For the most part, these subfields are optional within a field. Thus, it is possible to specify parts of a field without specifying the entire field.

When parameter fields and subfields are optional, the following rules apply, where an empty field is defined as one that contains no nonspace characters:

(1) Parameter field separators must be specified, left to right, through the last parameter given; fields preceding the last parameter may be empty; trailing field separators need not be specified.

(2) The same holds true of parameter subfield specifications within a field.

Leading spaces within a statement are permissible in the following cases:

- Following the at (@) character
- Following a colon (:) when a label is specified
- Following a parameter field separator (,)
- Following a parameter subfield separator (/)

A space, placed at any position in the coding other than those listed, is interpreted as the termination of the image.

In both batch and demand processing, data images and control statements in a run stream are processed sequentially and only upon request by the executive or by a program operating in that run.

However, a special mode of processing control statements is available during demand processing. This mode directs the executive to process a control statement immediately after it has been input from a remote inquiry terminal. The processing called for by the control statement is also done independently of any current program execution or control statement processing in the run stream. This mode of executing a control statement is specified by a special character, a second @, in column 2 on the control statement. This mode of operation is called transparent mode, and control statements which can direct or specify this mode of operation are called transparent control statements.

Transparent control statements are a subset of the control statement set. The syntax rules for normal control statements, with the following exceptions, also apply to transparent control statements. The exceptions are as follows:

(1) The identification of a transparent control statement consists of a @@ versus a @ for a normal control statement.

(2) The use of a label on a transparent control statement, while not prohibited, is meaningless.

The following discussion concerns the conventions for file or element names, the most common contents of the parameters.

Although the distinction between filenames and element names is often evident from the context, there are many cases where a period must follow a filename, or it will be either not accepted, or incorrectly treated as an element name. Therefore, it is best to always specify the period, as shown below:

filename is used to indicate: [[qualifier] *] file[(F-cycle)] [/[read-key] [/write-key]].

eltname is used to indicate: [filename.] element[/version] [(element-cycle)]

Qualifier, file, element, and version names are 1–12 alphanumeric Fieldata characters ($ and – characters are also allowed). Keys have 1–6 characters from the entire Fieldata character set, excluding only space, comma, slash, period, and semicolon. F-cycles are numbered upward from 1 to 999; element cycles are numbered upward from 0.

When the qualifier is omitted, the project-id from the @RUN control statement is used, except in the special case where a leading asterisk appears before the filename and a qualifier has been previously furnished on a @QUAL statement. When the F-cycle or element-cycle number is omitted, the most recently created cycle is used.

When the filename portion of an eltname is omitted, the system usually assumes an implicit reference to the run's temporary program file, TPF$.

A relative F-cycle of (+1) must be used to distinguish a newly assigned "to be cataloged" file (see @ASG,C and U options) from an existing catalogued file of the same name. A relative F-cycle of (–3) would designate the fourth oldest file that was catalogued under the specified filename. Element cycles are referenced by their actual number, such as (0) or (6).

Each item (image) in a symbolic element has a cycle number that indicates to which element cycle it was entered, and, if deleted, a cycle number to indicate in which cycle the item was deleted. When an element is updated, the added items are given an entered cycle number one greater than the last cycle of the element and the deleted items are given a deleted cycle number one greater than the last cycle of the element.

When specifying a symbolic element for compilation or assembly, the user may select a specific update from a sequence of retained updates by referencing the proper cycle number as part of the element name.

A system-standard maximum of five consecutively numbered cycles may be retained in a symbolic element. This maximum may be set to any value needed (up to 63) for a particular element by the use of the @CYCLE control statement. As soon as the number of retained updates for an element exceeds the specified maximum, the update with the lowest numbered cycle is combined with the update having the next higher cycle number to create a new cycle which in effect becomes the oldest cycle of the element.

A particular cycle may be referenced by either an absolute or a relative cycle number. Absolute cycle numbers are unsigned integers in the range 0 to 62; however, since only a limited number of cycles are retained, the absolute cycle numbers used when referencing an element must be in the range of those absolute cycles retained. Relative cycle numbers are signed integers. If the relative cycle is given as –n, then absolute cycle r–n is referenced, where r is the most recent absolute cycle retained. If +n is used, then absolute cycle x+n is referenced, where x is the oldest absolute cycle retained. The use of relative cycle numbers makes it unnecessary to know the absolute cycle number of either the oldest or most recent cycle retained.

Since absolute cycle numbers may not be greater than 62, when absolute cycle 62 of an element is updated all retained cycles are renumbered. The renumbering assigns cycle 0 to the oldest cycle retained, 1 to the next oldest, and so forth.

SYSTEM DATA FILE (SDF) FORMAT

Data in an SDF-formatted file is recorded in variable length images, with each image being preceded by a control word which specifies image length and type. Images are of two general types.

(1) Control Images

(2) Data Images

Control images provide various file control information as is needed by the individual component processing the file. A control image is indicated by bit 35 being set in the control word preceding it. The initial image of every SDF-formatted file must be a label image which is defined by a 50_8 in bits 35-30 of the control word. The control word for SDF control images has the format:

S1	S2	S3	S4	S5	S6
c	*l*	*ft*	*p*		*ct*

where:

c A unique code indicating what information is contained in the control image. The possible octal values of *c* are:

040 – Bypass this image

041 – Unique READ$ file label image

042 – ASCII/Fieldata switch

043 – Special FORTRAN back space

050 – Label image

051 – Continuation of previous image

054 – End-of-reel

056 – Start of accounting information in print file

057 – Output symbiont position marker

060 – Print control image

070 – Punch control image

076 – Demand breakpoint EOF

077 – End-of-file

NOTE:

All information in the label block (C = 041, C = 050) must be in Fieldata even though the file is ASCII.

l The length in words of the following image.

ft Used only in label blocks and is the file type as follows:

C – Symbiont card file

F – FORTRAN library data file

I – Symbiont input file (created by @FILE control statement)

P – Symbiont print file

S – Symbolic element (generally created by SIRASM)

p Used only in the label block of symbiont files and is the part number of the file (that is, a count of the breakpoints performed on the file).

ct The code type of the following images (applicable only where C = 042 or 050):

0 – Fieldata

1 – ASCII

Any image in an SDF-formatted file whose control word does not have bit 35 set is a data image. The control word for an SDF data image has the format:

0	*l*	*n*	*ct*
35	34 24	23 6	5 0

where:

n Used by each component to contain special information. For nonsymbiont files the n field extends through S6.

STANDARD PROCESSOR CALL STATEMENT

@PROCESSOR, options **SI-eltname, RO-eltname, SO-eltname**

where:

SI – source input

RO – relocatable output

SO – source output

STANDARD PROCESSOR OPTIONS

A – Accept the results of processing even if errors are detected. In any case, do not error exit.

I – Initial insertion of a new source language input element from the control stream. The source language output (SO) parameter is never used, as the source language input (SI) parameter specifies the eltname to be given to the source language output.

L – Produce the most comprehensive print listing available for this processor.

N – Produce the most abbreviated print listing available for this processor.

P – Specifies that source language output should be in Fieldata code. Identifies card image input, if any, as being in Fieldata. (Compare with Q option.)

Q – Specifies that source language output should be in ASCII code. Identifies card image input, if any, as being in ASCII. (If neither P nor Q is specified, the code type of the existing source language input element, if any, is used. Otherwise, Fieldata is assumed.)

S – Produce a moderately comprehensive print listing.

U – Update an existing source language input (SI) element to the next higher element cycle, thus saving any source language corrections that are currently being applied to the source language input element.

W – List correction lines at the head of the printer listing (this is feasible only for a two-pass processor).

X – Take error exit if errors are detected, to inhibit further processing of the run.

When the I option is not specified for processors that accept source language input, it indicates that the source language input (SI) eltname is an existing element from which the symbolic images are to be taken. To correct this source language input, directions must be given as to which of the existing images are to be deleted, and where any new images being furnished from the run stream are to be merged with the existing images. A special minus or hyphen character (–, which can optionally be changed to some other character such as + by including a card image which contains, in columns 1–3, a –, a =, and the new character, as in: –=+) is normally used to identify cards in the source language input that direct these corrections by referencing decimal line numbers in the source language input element. These line numbers that are given on correction cards must be in ascending numerical order. To add images the correction card has –n to mean: insert after line n in the source language input element the noncorrection card(s) which directly follow this card in the run stream. To delete or replace an image or images, the correction card has –n,m which gives a range of line numbers to be deleted (such as –1,1 or –68,146) and directs that the noncorrection cards, if any, which directly follow this card in the run stream are to replace the deleted images. Any card images which are not preceded by a correction card are added at the very beginning of the element.

Note about partial line corrections:

In addition to the –n and –n,m correction card formats discussed earlier, –n,m– line range specification may be used for applying partial line corrections. For each card image in the existing element in the range n through m, there must be a corresponding parameter card following the –n,m– card. Parameter cards may be any of the following four types, where a slash (or any not otherwise used character) serves as a separator:

beginning-col-number,ending-col-number/replacement-characters/
/existing-character-string/replacement-characters/
/existing-character-string/replacement-characters
beginning-column-number/replacement-characters

In the case of the second two parameters, the replacement characters are space filled out to the end of the card. For example, the parameter 73/ could be used to strip sequence numbers from columns 73 thru 80, leaving spaces in their place. In all cases, the information to the left of the separator identifies the portion of the card image to be removed. The replacement characters, if any, are inserted beginning at the left of this location. In the case of the first two parameters, where it may occur that the replacement characters do not fit into the space of the removed characters, the right portion of the existing image is right or left shifted as necessary.

@label:

A label is an alphanumeric field of one to six characters, the first of which must be an alphabetic. It is preceded by @ and terminated by a colon. Multiple labels may point to the same control statement.

Unattached labels may stand alone in run streams. A @JUMP to an unattached label passes control downward in the run stream to the next control statement which can accept a label.

Except where noted, a label may be used.

@ACOB[,options] SI-eltname,RO-eltname[,SO-eltname]

Call the ASCII COBOL language processor.

Options include the standard processor options and also:

B – Double space listing.

C – List Library text from COPY verb. S or O option must also be present.

D – Output allocation listing.

E – Output all diagnostic messages.

F – Output all diagnostics except Remark and Warning. Overrides E option.

G – Compressed symbolic on input cards.

H – Input cards contain sequence numbers in columns 73-80.

J – Card input with compressed symbolic in columns 1-72 and sequence numbers in columns 73-80.

K – Check sequence numbers in columns 73-80.

M – Ignore MONITOR statements.

O – Output object listing.

R – Output cross reference listing.

T – Reverse DISPLAY and DISPLAY-1. Reverse Comp and Comp-4.

V – Indicates subprogram rather than a main program.

Y – All listings written in Fieldata with PRINT$, instead of ASCII with APRINT$ (except for 'W' option correction lines).

Z – Interpret compiler DEBUG and OPTIONS statements if debugging compiler used.

@ADD[,options] eltname-or-filename

Add to the run stream, at the location of this @ADD control statement, all of the data and/or control statements from a specified standard data file (SDF) format element or file.

Options:

D – Allows the insertion of files or elements when operating under the DATA or ELT,D processors.

E – After the last image in the added data has been read, a subsequent ER READ$ request returns an end-of-file status, as if there had been an @EOF control statement at the end of the added data.

L – Use in demand mode only. Will list all control statements encountered in the added file or element at the demand terminal until the run stream returns to the demand terminal input.

P – Print this @ADD control statement image.

@ALG[,options] SI-eltname,RO-eltname[,SO-eltname]

Call the ALGOL language processor.

Options include the standard processor options and also:

C – This is a program which uses the SIMULA extensions to ALGOL.

F – Allow code in columns 73-80.

M – Increase compiler table by octal 010000.

O – Compile array addressing in line, to speed up execution time.

P – Inhibit printing of begin, end and block diagnostics.

R – Remove subscript checking.

T – Print the timing for phase 1 and 2 of compilation.

Z – Delete the formation of runtime diagnostic information.

■ FASTRAND FORMATTED FILE @ASG CONTROL STATEMENT

**@ASG[,options] filename,type/reserve/granule/maximum/placement,pack-id–1/.../;
pack-id–n**

@@

Assign FASTRAND-formatted mass storage files.

All parameters on the @ASG control statement are optional except filename. See facilities status bits in Section 4 for definitions of possible rejection or warning bit codes.

Options for cataloguing:

C – Catalogues file if the run terminates normally. If the file is freed prior to termination, the file is catalogued at that time.

P – Used with C or U option to catalogue file as a public file. If omitted, file is catalogued as a private file.

R – Used with C or U option to catalogue file as a read-only file. The file can only be read or decatalogued.

U – Same as C option except that the file is catalogued at run termination (regardless of the manner of termination beyond this statement). The @FREE control statement causes cataloguing prior to the termination.

W – Used with C or U option to catalogue file as a write-only file. The file can only be written into.

Options for catalogued files:

A – Specifies that the file being assigned is currently catalogued.

D – Used with A option to delete catalogued file from the master file directory (decatalogue) if the run terminates normally, or when a @FREE control statement is encountered prior to run termination.

K – Same as D option except that the file is decatalogued at run termination regardless of the manner of termination. A @FREE control statement decatalogues the file prior to termination.

Q – Requests that this file assignment be honored even if the system has disabled the file.

X – Used with A option to specify that this run is to have exclusive use of the file until the run has terminated or the file is released by a @FREE control statement or exclusive use is released via @FREE,X.

Z – Specifies that this control card is not to cause a hold condition. Control will be returned immediately and the assignment will be rejected if a hold state would have resulted.

Option for temporary files:

T – Specifies that the file is to be assigned temporary and allows it to have a name the same as that of an unassigned catalogued file.

Checkpoint/Restart Options for Control of Catalogued FASTRAND Files:

B – Dumps the file as a part of any checkpoint. This option must be used with the E, H, or M options.

E – Reload the file if any other run has referenced the file since the checkpoint. If no other run has referenced the file since checkpoint, the currently existing file is used with no reload.

H – Reload the file only if no other run has referenced the file since checkpoint. If another run has referenced the file since checkpoint, the currently existing file is used with no reload.

M – If a catalogued file by this name exists when reloading, make the reloaded file available to this run as a temporary file. If it does not exist, the file will be reloaded and assigned to this run as a catalogued file.

type subfield:

FCS – FASTRAND mass storage simulated in Unitized Channel (1106/1108) or Extended Storage (1110).

F4 – FASTRAND-formatted mass storage simulated on FH-432 drum.

F17 – FASTRAND-formatted mass storage simulated on FH-1782.

F40 – FASTRAND-formatted mass storage simulated on 8440 disc.

F8 – FASTRAND-formatted mass storage simulated on FH-880 drum.

F24 – FASTRAND-formatted mass storage simulated on 8424 disc.

F14 – FASTRAND-formatted mass storage simulated on 8414 disc.

F2 – FASTRAND mass storage, Model II and Model III.

F60 – FASTRAND-formatted mass storage simulated on 8460 disc.

F – FASTRAND-formatted mass storage, type independent.

FASTRAND mass storage simulated on drum or disc has all the characteristics of a FASTRAND file except for sector padding on write functions.

When space is not available for specified device type, another type is substituted which satisfies the request.

reserve subfield:

An integer specifying the number of granules required by the file (not to exceed 262,143). This parameter should give a reasonable estimate of the space needed to create or update the file. The value used for a file update must include those granules already in use.

granule subfield:

TRK–One track (64 sectors = 1792 words)

POS–One position (64 tracks)

If omitted, TRK is assumed.

maximum subfield:

Specifies the maximum allowable length (in granules) of the file. Permissible values are as for the reserve parameter. When specified, this parameter overrides the system standard maximum specified at system generation. If omitted, the reserve parameter value or system standard is used, whichever is larger.

placement subfield:

Placement may be any of the following forms:

ls logical subsystem

lslu logical subsystem and unit

ls may be any of the letters A to Z.

lu may be any of the numbers 1 to 15. The executive attempts to place files assigned to this run with different logical subsystems specifications on different physical subsystems. The same applies to logical unit specification. This is used to optimize I/O operations.

pack-ids:

Specifies the removable disc packs required for the file. Pack-ids consist of from one to six characters of the set A–Z, 0–9. The pack-ids for catalogued files are recorded in the master file directory and need not be specified on reassignments. Pack-id is applicable only to removable discs. If omitted (disc equipment is requested), fixed disc is assumed.

General Notes:

The device type of a FASTRAND-formatted file can be changed to a new type when extending a file. To make the change, the file must be reassigned as it was previously assigned, but with a different equipment code (device type).

The following rules apply:

(1) The file must be currently assigned to the run when the @ASG control statement with the new device type is submitted.

(2) This feature does not apply to removable disc files.

■ **MAGNETIC TAPE @ASG CONTROL STATEMENT:**

@ASG[,options] filename,type/units/log/noise/MSA-trans/unit-trans/format,reel–1/;
reel–2.../reel–n,expiration-period.

@@

Assign a magnetic tape file.

All parameters on the @ASG control statement are optional except filename and type. The A,C,D,G,K,P,Q,R,T,U,W,Y and Z options have the same meaning as on the FASTRAND-formatted @ASG control statement.

The remaining options are:

B – Binary (translation not required).

E – Even parity (assumed when the I option is specified and translation is performed by software). Not recommended if the file manipulation is via Univac-supplied software.

H – High density (800 FPI) tape (not available for UNISERVO 12/16 nine-track if the hardware dual density feature does not exist on the unit).

I – Decimal (translation required). The translation of BCD to Fieldata on input and Fieldata to BCD on output is performed by hardware, if available. Otherwise, standard system conversion routines are used for translation. The E option is assumed when software performs translation. Software translate is not available on the UNISERVO 12/16/20.

L – Low density (200 FPI) tape (not available for nine-track subsystems).

M – Medium density (556 FPI) tape (not available for nine-track subsystems).

O – Odd Parity (assumed).

V – Density mode of 1600 FPI (UNISERVO 12/16/20 nine-track subsystems only).

Options for tape labeling:

F – Writes information on a labeled output tape to indicate that any reassignment of the tape requires only verification of reel number. Absence of the F option when creating a labeled output tape writes into the label blocks information which forces all subsequent assigns to use the same qualifier and filename that were used to create the tape. This causes reel number verification and confirmation that the correct file has been associated with the reel.

J – Specifies that the reel loaded must be an unlabeled tape.

NOTES:

(1) In the absence of overriding mode options on seven-track tape assignments, the H and O options are assumed. For these assignments, mode option V is invalid.

(2) For UNISERVO VI-C and VIII-C nine-track tape assignments, the H and O options are assumed.

(3) For UNISERVO 12/16/20 nine-track assignments, the V and O options are assumed. For nine-track tape assignments mode options, B,E,I,L, and M are invalid.

filename:

The function and use of this parameter is the same as that specified for the FASTRAND @ASG control statement.

type:

Specifies that the @ASG control statement is for a magnetic tape device and identifies the specific type of unit required. Permissible entries for this parameter are:

T tape unit, type independent.

C UNISERVO IV-C, VI-C, and VIII-C seven-track tape units.

CB UNISERVO IV-C, VI-C, and VIII-C seven-track units with hardware translate.

C9 UNISERVO VI-C and VIII-C nine-track tape units.

U UNISERVO VI-C, VIII-C, 12 and 16 seven-track tape units.

UB UNISERVO VI-C and VIII-C seven-track units with hardware translate.

U9 nine-track tape unit, density independent.

U9H nine-track tape unit, 800 FPI density.

U9V nine-track tape unit, 1600 FPI density.

2A UNISERVO II-A tape unit.

3A UNISERVO III-A tape unit.

4C UNISERVO IV-C tape unit.

6C UNISERVO VI-C seven-track tape unit.

8C UNISERVO VIII-C seven-track tape unit.

12 UNISERVO 12 seven-track tape unit.

16 UNISERVO 16 seven-track tape unit.

12N UNISERVO 12 nine-track tape unit.

16N UNISERVO 16 nine-track tape unit.

12D UNISERVO 12 dual density nine-track tape unit.

16D UNISERVO 16 dual density nine-track tape unit.

20N UNISERVO 20 nine-track tape unit.

units:

Specifies the number of tape units required, and may be integers 1 or 2. If omitted or a number other than 1 or 2 is specified, the executive assumes that one unit is required.

log:

Assigns a single letter indicating a logical channel. The executive attempts to assign all files with the same letter to the same physical channel and those with different letters to different channels.

noise:

Specifies an integer from 1 to 99 which overrides the standard system noise constant. If omitted, the standard system noise constant is assumed.

MSA-trans:

For UNISERVO 12/16/20 assignments only. Specifies the type of translator needed in the MSA. The MSA translator mnemonics are:

EBCDIC – Fieldata to or from EBCDIC

ASCII – Fieldata to or from ASCII

XSEBCD – XS-3 to or from EBCDIC

XSASCI – XS-3 to or from ASCII

OFF – Turns off translator if assign is from the master file directory and file was catalogued with a translator specification.

unit-trans:

For UNISERVO 12/16 seven-track assignments only. Specifies the type of translator needed in the control unit. The control unit translator mnemonics are:

BCD – EBCDIC to or from BCD

DC – three eight-bit bytes converted to or from four six-bit tape characters

OFF – Turns off translator if assign is made from the master file directory and the file was catalogued with a translator specification.

format:

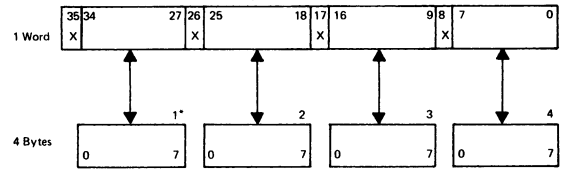
For UNISERVO 12/16/20 assignments only. Specifies the data transfer format for the word-to-byte conversion in the MSA. The data transfer mnemonics are:

Q – quarter word

6 – 6-bit packed

8 – 8-bit packed

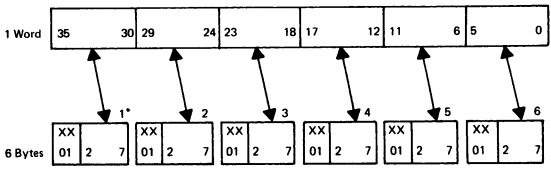
Quarter Word Format



NOTE:

Bits 35, 26, 17, and 8 are used for stop control on output operations and forced to binary 0 on input operations.

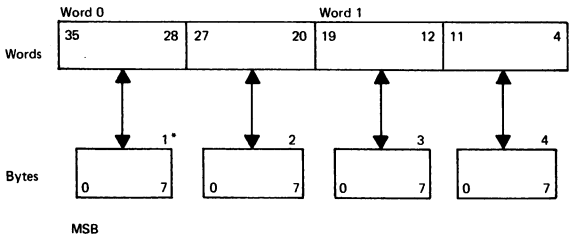
6-Bit Packed Format



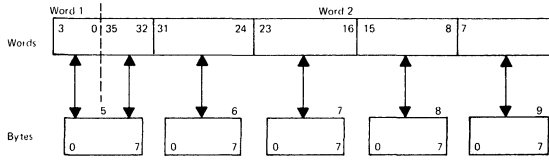
NOTE:

Bits 0 and 1 become binary 0 on output and are ignored on input, for each 8-bit byte. When translation is specified, bits 0 and 1 are not forced to binary 0.

8-Bit Packed Format



**Numbers on arrows indicate the order of byte transfer.*



reel:

Specifies the identifier for each tape reel required. Each reel identifier is limited to six characters from the set A–Z and 0–9. Reels are used and catalogued in order specified.

For temporary files, the reel parameter is not required, and the operator is requested to mount blank reels. If reel numbers are given on the @ASG control statement, they are used in the given order. When additional reels are requested, blanks are used.

expiration-period:

Specifies the number of days that this file is to be retained. The maximum number allowed is 4,095 days.

■ WORD-ADDRESSABLE MASS STORAGE @ASG CONTROL STATEMENT:

@ASG[,options] filename,type/reserve/granule/maximum,pack-id–1/pack-id–2/.../;
pack-id–n

@@

Assign word-addressable mass storage and simulated word-addressable mass storage.

All parameters are optional on the @ASG control statement except filename and type. With the exception of the following differences, the parameters of this statement are basically the same as those for the FASTRAND-formatted mass storage @ASG control statement.

Options:

Same as for FASTRAND-formatted mass storage except that no distinction is made between file types except in the conversion of logical to physical addresses.

type subfield:

Specifies that the @ASG control statement applies to word addressable drum format and names the specific type of recording equipment to be used. Permissible parameters are:

D	Word-addressable storage, type independent
D4	Word-addressable storage, FH-432 drum
D8	Word-addressable storage, FH-880 drum
D17	Word-addressable storage, FH-1782 drum
DCS	Word-addressable storage, Unitized Channel (1106/1108) or Extended Storage (1110)
D14	Word-addressable storage, simulated on 8414 disc
D24	Word-addressable storage, simulated on 8424 disc
D40	Word-addressable storage, simulated on 8440 disc

Use of the D entry is recommended since it allows the executive freedom in allocating file space.

The use of the D14, D24, or D40 entry forces the executive to simulate word addressability which introduces additional overhead each time the file is accessed.

Reserve:

Entry is in number of words.

Maximum:

Entry is in number of words.

Word addressable files cannot be used as program files. The mass storage allocation routine attempts to satisfy word addressable drum requests in the same manner as it satisfies FASTRAND-formatted requests.

■ ARBITRARY DEVICE @ASG CONTROL STATEMENT

@ASG filename,type,pack-id or reel
@@

Used for the assignment of special I/O devices, communications equipment, symbiont controlled devices, and online maintenance of all system peripherals. Symbiont devices and discs must be in a reserved state.

All parameters in the operand field are required, except pack-id or reel.

type:

(1) The mnemonic definition of a class of devices; the executive selects the specific unit if more than one unit exists.

(2) Absolute subsystem; the executive selects the specific unit.

(3) Absolute subsystem/unit.

For absolute subsystem assignment, the type parameter contains the subsystem number (1 to 127) in the format:

Sxxx

For absolute unit assignment, the type parameter contains both the subsystem number and the unit number in the format:

Sxxx/Uyy

If absolute subsystem and unit are used for communications devices, the unit specified must be the input rather than output or dial.

@ASM[,options] SI-eltname,RO-eltname[,SO-eltname]

Call the assembler.

Options include the standard processor options given earlier in this section (where S causes both the source language input and octal output to be printed), and also:

C Same as S option, except the octal output is not printed.

D Same as S option, except that a double-spaced listing is produced.

E Same as standard N option, except lines marked with error flags are printed.

F Identify the relocatable output code as quarter-word sensitive.

M Request that the assembler's internal symbol and procedure sample table area be expanded by an additional 10240 (10K) words. The M, R, and Z options may be combined.

O Print only the octal output.

R Request that the assembler's internal symbol and procedure sample table area be expanded by an additional 5120 (5K) words. The R, M, and Z options may be combined.

T Identify the relocatable output code as third-word sensitive. Compare this with the F option.

Y Minimize unneeded print (for Teletype*). Do not print '*new' or comments and shift PROC sample images to column 9. Must be used with other print option.

Z Request that the assembler's internal symbol and procedure sample table area be expanded by an additional 20480 (20K) words. The Z, M, and R options may be combined.

@BRKPT[,option] generic-name[/part-name]
@@

Close out the currently active file part as identified by generic-name, and start a new part as identified by part-name. The @BRKPT command may be applied equally to standard PRINT$ or PUNCH$ files, either EXEC or user-defined, or to read, print or punch alternate files.

**Trademark of Teletype Corporation*

Options:

E – Inhibits EOF positioning for alternate read files on magnetic tape. For input tapes, the @BRKPT command is normally used to prematurely terminate reading of the file; in the absence of the E option the input tape is positioned forward to the next EOF mark to allow reading of a subsequent file on the tape. The use of the E option avoids needless tape movement when the user is finished with the tape.

L – Used to provide a method of stacking multiple files on a single magnetic tape, and provide a label for each file by which it may be referenced on a subsequent @SYM command. The label is the part-name from the @BRKPT,L command(s).

generic-name:

Identifies the file part to be closed. May be either the name of a user defined file or the generic name PRINT$ or PUNCH$ if it refers to a standard output file. If the generic-name refers to a standard output file, its contents are automatically queued for output at the appropriate peripheral device. If the generic-name refers to a user-defined file, the file is closed by writing an EOF mark. No attempt is made to queue user-defined output files (requires use of the @SYM control statement), and in the case of a magnetic tape file, the tape is positioned such that a new file may be started.

part-name:

Identifies the new file part. Is only required if the new part is a user-defined file. If the part being closed is a standard output file (generic-name is PRINT$ or PUNCH$) and the new part is also to be a standard output file (not user-defined) the part-name field need not be included in the @BRKPT statement.

(1) The only legal punctuation mark on a @BRKPT control statement is a (/). A period following either the generic-name or part name is illegal.

(2) A catalogued file must be used for a user-defined file if it is to be printed or punched later (see @SYM control statement).

(3) In normal demand operation there is no PRINT$ file and therefore, a '@BRKPT PRINT$' command is illegal unless the user has previously submitted a '@BRKPT PRINT$/user-file' command.

(4) A user-defined mass storage file should be used as a part-name only once unless a means is employed to save the file data, since attempts to write multiple parts into such files causes overwriting of previous parts.

@CAT[,options] operand
@@

Directly catalog a mass storage or tape file in the master file directory, without first assigning facilities to the run.

Operand:

This has the same operand field as @ASG for mass storage, or @ASG for tape except that expiration period is not used. All parameters on the @CAT control statement are optional except filename.

Options:

B,E,G,H,I,J,L,M,O,P,R,V, and W as described in magnetic tape and FASTRAND-format or word-addressable file @ASG.

See facilities status bits in Section 4 for definitions of possible rejection or warning bit codes.

@CHG[,options] eltname-or-filename[with keys],
eltname[-or-filename[with keys]]

Call the FURPUR processor to: a) change the name of a program file element, or, b) to change a catalogued file's name, modes and/or keys.

Options for a) are A,C,O,R, and S as defined under @FIND. Any combination of these options may be used.

Options for b) are:

P – set public mode.

Q – set private mode.

V – set read-only mode, clear write-only mode.

W – set write-only mode, clear read-only mode.

Z – clear read-only or write-only mode (must not be used in conjunction with V and W options).

@CKPAR[,options] filename.element
@@

Establish a program checkpoint dump that may be used for restart at some future time.

Options are P and T as defined under @CKPT.

filename:

Specifies the user-assigned FASTRAND-formatted file currently assigned to the run.

element:

Specifies the name to be given to the element created by the checkpoint. Only one checkpoint per element is permitted.

NOTE:

The @CKPAR control statement has no meaning within a run stream because there is no program executing at the time the statement is encountered.

The error status codes returned by @CKPAR are displayed in the description of @CKPT.

@CKPT[,options] filename

Produce a checkpoint dump of this run.

Options:

P – The dump completion message and all error code messages are to be written to the console.

T – Terminate this run after this checkpoint is taken.

The specified filename must be assigned and may be either a tape or mass storage file which is catalogued public. While only the last checkpoint directed to a FASTRAND mass storage file is saved, all of a run's checkpoints that are directed to a tape file are saved.

The following are checkpoint error status codes (which appear in register A0 if @CKPT is called via ER CSF$). They apply to complete and partial checkpoint as indicated in the table.

Error Code (octal)	Description
01	Unrecoverable magnetic tape error.(2)
02	Unrecoverable FASTRAND-formatted mass storage error. (1,2)
03	Checkpoint file not assigned, filename specified for a partial checkpoint is not a program file, or unable to assign checkpoint file if request was from keyin. (1,2)
04	Checkpoint requested on illegal device type. (1,2)
05	Read or write is inhibited on the FASTRAND-formatted checkpoint file. (1,2)
06	Lost position on tape (other than checkpoint file). (2)
07	Run has non-write-protected common bank. (1,2)
010	The PCT is at its maximum size and cannot be expanded. (2)
011	The run is already being checkpointed. (1,2)

NOTES:

(1) May occur during partial checkpoint.

(2) May occur during complete checkpoint.

Error Code (octal)	Description
012	A program is executing in real time (for keyin only), or ESI activities are present or the run is demand, or the run has write enabled common banks attached. (1,2)
013	Format error. (2)
014	User's program file is not assigned. (1)
015	Checkpoint file is not large enough – I/O status of 22_8 status was returned. (1,2)
017	Transparent request of run having no activities. (1)

NOTES:

(1) May occur during partial checkpoint.

(2) May occur during complete checkpoint.

@CLOSE[,options] filename,. . .

This has the effect of a @MARK, followed by a @REWIND.

Options:

C – Continue. Do not error terminate, even if errors are detected.

I – Rewind with interlock.

@COB[,options] SI-eltname,RO-eltname[,SO-eltname]

Call the COBOL language processor.

Options include the standard processor options given earlier (where S causes only the symbolic input to be listed, and L is the same as if the C,D,E,K,O,R and S options had been specified). Also:

B – Ignore sequence number check.

C – List matched names of CORRESPONDING data names.

D – List data definitions, with qualification.

E – List detailed error diagnostics.

F – Causes COB$PF file to be searched first when attempting to satisfy COPY statement.

G – Input is compressed symbolic in columns 1-80.

H – Input contains sequence numbers in columns 73-80.

J – Input is compressed symbolic in columns 1-72 and sequence numbers in columns 73-80.

K – List all parts incorporated by the COPY and INCLUDE verbs.

M – List source language COPYed.

O – List octal output of final phase.

R – List cross references, excepting qualified names.

T – Demand terminal use of ACCEPT and DISPLAY verbs is permitted.

V – Indicates subprogram rather than a main program, and prevents generation of a starting address.

Y – Causes COB$PF file to be searched after source input file when attempting to satisfy COPY statement.

Z – Source line/relative address cross-reference.

@COL CB,sentinel

Permits the user to switch read mode to column binary. The @COL control statement is only valid when read from an onsite card reader. Three blank cards are needed after the @COL CB card.

Only the sentinel parameter is optional in the @COL CB control statement. A label is not allowed.

sentinel:

Specifies user-defined sentinel for terminating the nonstandard read mode data input stream. The sentinel may consist of from one to five characters.

@COL Command

Permits the user to switch to a read/translate mode at the onsite 9000 card reader that is other than the system standard.

Command specifies the mode to be used for the next, and succeeding card reads until a new @COL control statement (or a @FIN) is read. No sentinel is needed. The @COL control statement may be used at any point in a run stream or may precede the @RUN control statement if it is necessary to establish a new input mode for a specific card deck.

The commands are as follows:

1100FD	Read 1100 cards (FD), send Fieldata images.
9000FD	Read EBCDIC cards, send Fieldata images.
1100AS	Read 1100 cards (FD), send ASCII images.
9000AS	Read EBCDIC cards, send ASCII images.
ASCFD	Read ASCII cards, send Fieldata images.
ASCASC	Read ASCII cards, send ASCII images.

No blank cards are needed after these @COL control statements.

@COPIN[,options] input-filename-or-eltname[,output-filename-or-eltname]

This is essentially the reverse of @COPOUT, where elements that were stored on tape in @COPOUT (element file) format are added back into a program file on mass storage.

Options:

A,O,R and S – Same definitions as for @COPY, except that the input filename (or the filename portion of an input eltname) identifies a tape file.

C and V – Same definitions as for @COPOUT.

When a particular element is named for @COPIN, only one of the A,O,R, or S options may be specified.

@COPIN used without options must specify only filenames. This causes all elements up to the next end-of-file (EOF) mark on the specified tape file to be copied.

If the second parameter is omitted, the element name in the first parameter (if any) is used as the output element name; the output file in this case is assumed to be TPF$.

@COPOUT[,options] input-filename-or-eltname,output-filename-or-eltname

Call the FURPUR processor to copy onto tape a whole program file, or individual elements from within a program file.

Options:

A,O,R and S – Same definitions as for @COPY, except that the output filename (or the filename portion of an output eltname) identifies a tape file.

C – Continue. Do not error terminate, even if errors are detected.

V – Copy only elements of the version name indicated in the first parameter. If no version name is given in the first parameter, copy only elements with no version name. If a version name is given in the second parameter, attach this version name to all copied elements. A format of filename.[/version] is used in each parameter. The A,O,R, and S options may also be used to limit the copying to elements of the specified type(s).

If A,O,R,S, or V options are used on one or more @COPOUT control statements, a final @MARK should be done.

@COPOUT without options copies an entire program file to tape, and does an automatic @MARK operation.

@COPY[,options] input-filename-or-eltname,output-filename-or-eltname[,M-count]

Call the FURPUR processor to copy a file; or to make copies of the elements in one program file, and add them to another program file.

M-count is used with @COPY,M on a tape-to-tape copy and @COPY,B.

Options:

A – Used only with program files. If input and output eltnames are specified, copy a single absolute element into the output file, under the given output element name (if no output element name is given, the input element name is used). If whole files are specified, all nondeleted absolute elements in input filename are copied and added to output filename.

B – Used with 1100 Series FORTRAN files only. Copy the number of files (FORTRAN) specified in M-count from the input file to the output file. The input and output files cannot both be on mass storage or both be tape files.

C – Continue. Do not error terminate, even if errors are detected.

F – Copy contents of one file into another file. Program and element files must not be copied. Input file must be in SDF format. Input reading is terminated by the SDF EOF. Block size for tape files must be 224 words. When output is to magnetic tape, two hardware EOF marks are written following the file and the tape is positioned between the EOF marks.

G – This is used to save a copy of a mass storage file on tape (in which case the M option is also used to perform a @MARK operation). Files saved in this manner are also restored to mass storage by using @COPY,G. @COPY,G does track size transfers of data to or from mass storage, without regard to the file's original format.

I – Copy an SDF file, named as input filename, and add it to a program file as a symbolic element, specified as output eltname.

M – Except as noted elsewhere under @COPY options, this is used without other options for copying "files" (see definition of "file" under @MOVE) from one tape to another, to specify that a @MARK is to be done after each "file" is written to the output tape. The third parameter on a @COPY,M command may be used to state the number of "files" to be copied, which is assumed to be one if not stated. @COPY,M stops if two adjoining EOF marks are encountered, which is interpreted as the end of the tape.

N – Copy a tape containing an abnormal frame count to another tape file or to a FASTRAND-formatted mass storage file.

O – Same as A, except for omnibus elements.

P – The input and output filenames must be program files. This copies all nondeleted elements from one file and adds them to the other file.

R – Same as A, except for relocatable elements.

S – Same as A, except for source language elements, including proc elements.

@COPY without options may be used with two mass storage files of any format, to overwrite output-filename with the contents of input-filename.

@CULL[,options] [ASMorDATA/stop-column(pos)] [,eltname-or-filename,...]

Call the CULL processor to produce an alphabetically sorted, cross-referenced listing of all symbols in a specified set of source language elements. A symbol may contain up to 12 alphanumeric or $ characters.

Options:

A – Exclude symbols beginning with an alphabetic character.

C – Condense the printout listing to fit on a 72-column page.

D – Reverse the normal rule for including symbols that begin with a number or $. (In DATA mode, where everything is normally included, this will cause these symbols to be excluded. In ASM mode, where numbers and the free-standing $ character are normally excluded, this will cause them to be included.)

E – Inhibit all page ejects.

I – Used in ASM mode to add all the common assembler mnemonics for instructions, j-designators, registers, etc., to the exception table list.

L – In addition to performing any requested CULL operations, list the source language text of every element being processed, and identify the images that were inserted in producing update cycles of the element.

M – Reverse the normal meaning of the exception symbol list, and include only the symbols which appear in this list.

N – Used in ASM mode to reverse the meanings of the U,W, and X options, so that the symbols used as labels are included, but nothing else is.

O – Exclude all elements that are not proc elements.

P – Exclude proc elements.

Q – Used in ASM mode to cause symbols inside quote marks (which are normally excluded) to be included.

S – Print out the exception table list.

U – Used in ASM mode to omit referenced labels.

W – Used in ASM mode to exclude all symbols that are used anywhere as a label (which means that they are defined).

X – Used in ASM mode to exclude all symbols that are used anywhere as a label that is followed by an asterisk * (which means they are external definitions).

Y – Used in ASM mode to cause the W and X options to be ignored for elements not explicitly named in the @CULL eltname list.

Z – Used in ASM mode to include source language elements which have a processor code of 0, meaning "unmarked" (see ER PFI$ – Section 2). This is sometimes necessary for @ASM assembler source language elements which are either very old, or have been worked on by other processors.

The CULL modes are ASM and DATA, where ASM is assumed if none is stated. The stop-column is the last column which is examined in scanning each image (symbols are truncated if they extend beyond this column). If the stop-column is not stated, it is assumed to be 40 (to exclude the comment field) in ASM mode, or 80 (the full card image) in DATA mode.

(pos) is the number of positions to be reserved for @CULL's scratch file. It is assumed to be two if not stated, and has to be a larger value than two only if the number of symbol references is expected to possibly exceed 70,000.

A list of eltname and/or filename parameters may be given. Otherwise, all source language elements in TPF$ are processed.

One or more data cards may directly follow the @CULL control statement to provide an exception table of symbols to be excluded. As many symbols as will fit, separated by spaces, may be listed on each of these cards.

In the listing for an ASM mode CULL, the line numbers where a symbol is defined are flagged with an * (or **, if the label itself was followed by an *). The line numbers where a symbol is used as a directive (proc call, instruction mnemonic, etc.) are flagged with a D.

@CYCLE eltname-or-filename[,maximum-number-of-cycles]

Call the FURPUR processor to set a new limit on the maximum number of cycles to be retained by a source language element or catalogued file.

If the number of cycles currently retained is greater than the new maximum, a new element is created with the necessary oldest cycles deleted. Deleting an old element cycle involves dropping only those card images that were marked deleted when that cycle was produced.

When a catalogued file is named, the number sets the maximum range of F-cycles for the file. The file specified must be in the master file directory. If n is 0, the F-cycle series is deleted. If n specifies a new maximum less than the current range of F-cycles being retained, enough F-cycles of the file set (starting with the oldest cycles) are deleted to satisfy the new range.

If the 'maximum-number-of-cycles' field is omitted, current cycle information is listed for the filename specified.

@DATA[,options] input-filename[,[output-filename] [,sentinel-not-exceeding-6-chars]]

Call the DATA processor to transfer source language card images into an SDF format file.

Options:

I – Overwrite any existing information in input-filename with the source language images which follow in the run stream. The output-filename parameter is not used.

L – Produce a listing.

Any file-name referenced on a DATA call statement must have been previously assigned to the run and/or catalogued.

If neither the I option is on, nor output-filename specified, the L option is assumed, and input-filename is listed.

DATA does not write a tape mark after output of a file to tape.

@DELETE[,options] eltname-or-filename,...

Call the FURPUR processor to mark an element in a program file as deleted, or to drop a catalogued file from the master file directory.

Any combination of the options A, O, R, S and C may be used, as defined under @FIND. If neither A, O, R, nor S is used, one or more presently catalogued files may be named.

@ED[,options] SI-eltname-or-filename[,SO-eltname-or-filename]

Call the text editor processor to add new images, or modify existing images, in a source language element or SDF format file.

Options:

A – An auto-recovery is to be attempted.

B – Batch mode operation in demand environment.

D – Demand mode operation in batch environment.

E – Echo input lines.

I – A new element or file is to be built. Enter the editor in input mode.

N – Enter editor in brief mode.

P – Output is converted to Fieldata.

Q – Output is converted to ASCII.

R – Read file only – no corrections to be applied.

U – An existing element or file is to be updated.

■ **ED Processor Commands**

COMMAND	DESCRIPTION
ADD name or ADD name num1 num2	This command is used to add all or portions of a file to the current file. The first form adds the whole file, and the second form adds lines 'num1' through 'num2' to the current file. The lines to be added are inserted at the end of the file unless a + immediately follows the command in which case the lines are inserted following the current position within the edit file. The 'name' is the element or filename.

COMMAND	DESCRIPTION
APPEND	Go to the end of the element or file and enter input mode thereby allowing new images to be inserted.
AUTO num1	This command specifies that an automatic save of the current file is to be performed in case of processor or system failure. The 'num1' specifies that the auto save is to be performed for every 'num1' input transaction.
CASE UPPER CASE NORMAL	CASE UPPER causes all input lines to be translated to upper case. In CASE NORMAL mode, no translation is made. CASE UPPER is assumed for Fieldata files or elements.
CCHAR char	This command sets the continuation character. When an input line to the editor has this character in it, the editor assumes that the next line of input is a continuation of this current line. To return to the original continuation character (set by the system), use this command with no 'char'.
CHANGE/string–1/string–2/ m G or C/string1/string2/ m G	This command searches a specified number of text lines for 'string–1' and substitutes 'string–2' for it. The number of lines to be scanned is indicated by 'm'. The global indicator, 'G' changes all occurrences of 'string–1' in the range of lines. Any character besides G (or no character) changes just the first occurrence. The '/' (slash) may be any character which does not occur in 'string–1' or 'string–2' except a blank. If 'm' is omitted, 1 is assumed. The user may change all subsequent occurrences in the file by using the word 'ALL' (abbr. A) where 'm' is specified.
CLIMIT column number	This command allows the user to set a limit on the number of columns which will be searched during performance of the change command. The default value is 132.
CSF executive control statement	This command is used to submit a control statement via CSF$. Only statements valid for CSF$ may be submitted. The control statement must start in column 5.
CPT	This command prints out the SUPs used so far in the present run.
CPUNCH num1 num2 CPUNCH num1 CPUNCH	This command is used to punch parts or all of a file at an onsite card punch. After the command is entered, a message as follows will be typed out: MSG? The line typed in will be sent to the system console before the cards are punched.
DELETE num1 num2 D num1 num2 DELETE num1 D num1	This command is used to delete lines from the text. The first form deletes lines 'num1' through 'num2'. The second form deletes the next 'num1' lines starting with the current one.
DITTO num1 DITTO num1 num2	This command allows duplication of other lines in the file. The duplicated lines are inserted at the present position in the file. The first form results in the one line at 'num1' being inserted in the present position. The second form results in all lines 'num1' through 'num2' being duplicated at the present position.

COMMAND	DESCRIPTION
EXCH char octal number	This command is used to allow input of characters not represented in the keyboard character set. 'char' is the character which is to be used to stand for the number whose internal representation is 'octal number'. When 'char' occurs any place in an input line it will be replaced by this character. An EXCH with no parameters disables this feature.
EXIT	This is the command used to take a normal exit from the ED processor. All the corrections will be applied to the designated file and a normal exit will be taken.
FIND mask	FIND searches for an image which corresponds exactly column for column starting at column 1 with the 'mask'. Transparent characters may be in the mask which will test successfully with any character in the column. The search begins with the line following the current one and proceeds until a match or end-of-file is detected.
FC mask	The FC command behaves in the same way as the FIND command except that all occurrences are flagged in the remainder of the file.
IB string	This command behaves exactly the same as the INSERT command except that the line is inserted before instead of after the current line.
INLINE number term-sub	This command allows inline editing of a given line. If 'number' is blank, the current line is assumed to be the one to be edited. Otherwise the editor proceeds to line 'number'. The line will be printed out. The user can then enter editing information directly below the line to modify it. Following are the editing characters to be used. I – The string following this command is inserted following the character immediately above the I. The string is delimited on the right by the termination character '!'. R – The characters following the R will replace the characters immediately above them. A ! is required to terminate replacement. D – The characters in the line above are deleted between the D and the !. An alternate termination symbol may be specified as 'term-sub'. This will remain in effect for this INLINE only.
INSERT string I string	This command is used to insert a line following the line presently pointed at by the editor. This new line will then be the point at which the editor is positioned. The string to be inserted starts after the first blank following INSERT. If a '+' immediately follows the command, the string may be input on the next line. If the command with no image is entered when not in EOF mode (see 'ON' command) the editor will switch to input mode. In EOF mode this simply results in the insertion of a blank line.
INPUT	This command directs the editor to enter a special input mode. In this mode everything which is typed in is inserted in the file until an exit from the mode is taken. Exiting from this mode is accomplished by typing an @EOF when in EOF mode (see ON and OFF commands) or a carriage return when not. Tabs are recognized in this mode.

COMMAND	DESCRIPTION
LAST	This command directs the editor to move to the last line in the file and stay in edit mode. The last line may not be altered at this time.
LNPRINT num1 num2 LNPRINT num1 LNPRINT! LNP	This command behaves like the PRINT command except that each line is preceded with its line number. Syntax is the same as the PRINT command.
LNQUICK num1 num2 LNQUICK num1 LNQUICK! LNQ	This command behaves like the QUICK command except that each line is preceded with its line number. Syntax is the same as the QUICK command.
LNSITE num1 num2 LNSITE num1 LNSITE!	This command behaves the same as the SITE command except that each line is preceded with its line number.
LOCATE string LOCATE quote-char string quote-char	This command is used to search the text for a given string of characters. The search begins at the line following the current line and proceeds sequentially through the text until a find is made or the end of file is encountered. The first form ignores multiple blanks in the images. The second form requires that the text image be exactly the same as the string within the two quote characters.
LC string LC quote-char string quote-char	LC behaves as LOCATE except that all occurrences of the string in the remaining text are located. Just before each line containing an occurrence is typed out, the line number is typed out.
LCHAR char	This command sets the quote character for the LOCATE command. The default character is quote ('). A non-input character will be assumed if 'char' is a blank.
MAIL user-id	This allows the user to send messages to another user. The user-id is the original run-id of the person to whom the message is directed. The editor will then solicit 10 lines of input with: MAIL** If the desired message is to be less than 10 lines, the mode can be ended by entering an @EOF. After the message is received by the designated person it will be deleted.
MAXLINE number	This sets the maximum length to which a line may increase. If it is exceeded, the line will be truncated. The default is 80.
MOVE num1 MOVE num1 num2	This command performs the same operation as the DITTO command except that the original lines are deleted after the duplication has taken place. The syntax is the same as for the DITTO command.
MSCHAR char	This command sets a character which will be translated to a master space when it is input in column one. If 'char' is a blank, no master space translation is available.
OMIT	This is the command to be used if the user does not want his corrections to be applied to the file on exit. The input file will remain as it was at the beginning of the editing session, and the output file, if any, will not be produced.
ON special mode,...,special mode OFF special mode,...,special mode	This command is used to define some special modes within the editor. ON turns the mode on, and OFF turns it off. The special modes which may be abbreviated to one letter are: QUICK – compress extra blanks out of all output to device.

COMMAND	DESCRIPTION
	BRIEF – do not echo corrected images for CHANGE and DITTO. NUMBER – precede each line printed out with its line number. PCNTRL – recognize and print print control images. DSPLIT – delete lines transferred by SPLIT command. XBRIEF – do not echo lines transferred by SPLIT or ADD commands. SEQ – print sequence numbers when soliciting input. LOOK – look for mail after each command is executed. EOF – special mode where blank lines may be entered. INP command enters input mode and @EOF exits from input mode to edit mode. While in input mode blank lines may be entered. Also the INSERT command with no image following will enter a blank line. MEMORY – remember modes on successive executions. UNISCP – allow correct character placement on UNISCOPE with the INLINE command.
OPR string OPR* string	This command is used to send a message to the system console. The first form sends the message 'string'. The second form does the same, but also solicits an answer. The string may not be more than 50 characters or it will be truncated.
PCN	This command is used to enter a print control image into the file being edited. When the command is entered, the editor will solicit the image with: CONTROL IMAGE– This image can only be read when in a special mode set by the ON command.
PLIMIT column number1, column number2	This command is used to set left and right limits respectively on the columns printed out by the PRINT command.
PRINT num1 num2 PRINT num1 PRINT! P	This command is used to print out lines of text. The first form prints lines 'num1' through 'num2'. The second form prints the next 'num1' lines. If the command is immediately followed with a + the printing starts with the next line instead of the current one (example: PRINT+3). The third form prints the entire file from the top. If no number or recognizable symbol follows the command, a 1 is assumed and the present line will be printed out.
PUNCH num1 num2 PUNCH num1 PUNCH	This command is used to punch paper tape for form II paper tape input at a terminal which has punch and read hardware. The syntax for this command is the same as that for the PRINT command. When the command is entered, the following response will be given: DEPRESS PUNCH ON The processor will then pause to allow the user to push the punch-on button on the paper tape punch hardware. After pausing, the designated lines will be typed out which will cause the paper tape to be punched at the same time. Rubouts will be

COMMAND	DESCRIPTION
	punched at the start and finish of the tape. The tape so produced can be used as normal form II input.
QUICK num1 num2 QUICK num1 QUICK! Q	This command prints lines with all nonsignificant blanks omitted. 'num1' and 'num2' are the same as on the PRINT command. Plus (+) may also be used on the second form with the same meaning.
RETYPE string R string	This command is used to completely replace the current line with the string following the first blank after the command. A + may be used after the command with the same meaning as with the INSERT command.
RP number	This command is used to set a repeat counter for the INSERT command. Any insertion will be repeated 'number' times.
SCALE number	This command causes a line to be printed out which can be used for column sensitive operations. The form of the line is: 1234567890123456789012345678901234567890... starting in column 'number'.
SEQ.id i,j	This command causes sequencing to be inserted in columns 73-80. The starting value is 'i' and 'j' is the increment. The id overlays the numbers from the left.
SET tab1 tab2 tab3 ... tabn	This command is used to set the tabs for the commands which allow them as explained above. As many tabs as desired may be designated. Each SET command redefines all previous tabs, and so a SET with no tabs clears the tabs. If no SET has been performed a default of 11,21,39,73 is assumed.
SITE num1 num2 SITE num1 SITE	This command is used to direct output to an onsite printer (PR). The meanings of 'num1' and 'num2' are the same as for PRINT except that if no numbers are given, the third form is assumed. After this command is entered, a message as follows will be typed out: HDG? The line typed in will be used to head the onsite output. Periods must not be used in this header as anything beyond the period will not be printed. After the output is done, the following will be typed: MSG? The user should enter the information necessary to indicate where and to whom the output should be returned.
SPLIT name SPLIT name num1 num2	This command is used to build new elements or files from portions of a current file. The first form causes all the lines preceding the line currently pointed at to be reproduced as the designated file. The second form causes lines 'num1' through 'num2' to be reproduced. An '!' immediately after the SPLIT command causes the whole file to be copied.
STATUS special mode,...,special mode	This command is used to request the status of special modes set by the ON and OFF commands. If no special modes are specified, the status of all will be listed.
TAB tab-char	This command is used to specify which character is to be used as a tabulator character. This character is recognized on the INSERT, IB, and RETYPE strings and is recognized on all input when in the input mode. The character is not transmitted to the file and

COMMAND	DESCRIPTION
	behaves just as a tab on a typewriter. If no character is specified, a semicolon (;) is the tab character.
TCHAR char	This command sets a transparent character for the FIND command. No char disables the feature. TCHAR BL resets it default of blank.
TIME	This command prints out the date, time and cycle information and the name of the output element or file.
TYPE processor-mnemonic	Sets the processor type for symbolic element output. The processor mnemonics are: ALG, APL, ASM, COB, DOC, ELT, FOR, LSP, MAP, BAS, SSG and SEC. Octal numbers may also be used instead of the mnemonic.
UP	This command is used to cause an element or file to be saved as if the U was specified on the control statement. This is used if the entry to the editor was made with an R option.
number +number −number	These commands are used to position the editor at a desired line in the text. The first form directs the editor to line 'number'. The second form directs the editor to move to the position current line plus number. The third form directs the editor to move to the position current line minus 'number'. When the specified line is located, it is typed out if not in BRIEF mode, and modifications may be made to it. If it is desired to insert lines before line 1, 0 may be typed in. This will position the editor immediately before the first line.

@ELT[,options] SI-eltname[,[SO-eltname] [,sentinel-not-exceeding-6-chars]]

Call the ELT processor to insert or update an element in a program file. @ELT,D is used to create run streams beginning with a @RUN for use with @START, or partial run streams for use with @ADD.

Options include I, L, and U, as defined earlier under standard processor options; A, R, and S as defined under @LIST; and also:

D — Used instead of the S option to indicate that the source language input images following the @ELT control statement may include one or more control statements (@ in the first column) which should be transferred as data, rather than immediately processed as control statements. When the D option is used, all control statements documented in this manual (except @END and @ENDCL) that follow the @ELT statement are transferred, until an @END or @FIN is encountered.

Pairs of @ELT,D and @END control statements may be nested by using a different sentinel with each pair. A sentinel is any string of one to six alphanumeric characters.

The A and R options are used only with the I option, for introducing an absolute or relocatable element into a program file from card images in the run stream (see @PCH).

Source language elements (denoted by @ELT,S or @ELT,D) may be updated using the same correction conventions as noted under @ASM.

@ENABLE filename,...

Call the FURPUR processor to remove a "disabled" flag from a catalogued file.

@END [sentinel-not-exceeding-6-chars]

Notify system that this is the end of control stream images that are to be transferred as data by the previous @ELT,D or @DATA control statement. A label is not allowed.

@ENDCL

Terminate mode established by @COL CB control statement. Three blank cards are necessary after the @ENDCL card. A label is not allowed.

@ENDF

Mark the end of the images for a file created by the @FILE control statement. A label is not allowed.

@ENDX

When encountered on an ER READ$ while in CLIST mode (see ER CLIST$ – Section 2), will cause return with CLIST index of 77_8 and terminates CLIST$ mode. A label is not allowed.

@EOF [sentinel-character]

When encountered on an ER READ$, will give an end-of-file (EOF) return, which is useful for separating data in the control stream. The sentinel character may appear only in column 6. A label is not allowed.

@ERS filename,...

Call the FURPUR processor to return to the system all FASTRAND-formatted mass storage granules allocated to a file.

@FILE (not available for demand users)

Create SDF-formatted mass storage or magnetic tape files while the input symbiont is reading the run stream.

For each storage device, the format of the @FILE control statement is identical to the @ASG control statement for that device (except that the label field is not allowed). The file into which the images are placed may be either a FASTRAND-formatted or magnetic tape file. FASTRAND-formatted files may not be temporary files. The @FILE statement is valid only within a @RUN.

@FILE control statement processing is terminated upon encountering an @ENDF control statement, a @FIN control statement, or another @FILE control statement. Data images and all control statements except @COL and its accompanying end sentinel are placed into the created file (the @COL control statement and sentinel are processed immediately, and the file is marked when the mode switch is made).

Separate files may be created by separate @FILE control statements. However, if the current @FILE is being written to tape and the next @FILE control statement has the same filename and qualifier, the current file will be closed, end-of-file will be written on tape, and the second file will follow. Otherwise, a separate file will be created as specified on the current @FILE card.

@FIN

Terminate a run.

This control statement is always processed. It cannot be passed as a data image under @ELT,D or @DATA. A label is not allowed.

@FIND,option[s] eltname

This is normally used prior to a single-element @COPIN, to position a tape file in @COPOUT (element file) format to the element within the file which is to have the @COPIN done on it.

Options:

A,O,R, or S – Indicates an absolute, an omnibus, a relocatable, or a source language element.

C – Do not error terminate even if errors are detected.

Prior to using @FIND, the tape must be positioned to the "file" (where "file" is as defined under @MOVE) in which the requested element resides. The filename portion of eltname is that of the tape file.

@FOR[,options] SI-eltname,RO-eltname[,SO-eltname]

Call the FORTRAN language processor.

Options include all of the standard processor options given at the beginning of this section where the N listing option is assumed if neither L,N, nor S are given.

@FREE[,options] filename
@@

Release the physical facilities assigned to this run under the specified filename. Options may specify changing the characteristics of filename's assignment.

Options:

A – Release only the @USE-attached name specified in filename. Take no other action unless indicated by other options.

B – Release the @USE-attached name specified in filename. If this is the only @USE-attached name, do a normal @FREE of the file.

D – Drop this catalogued file from the master file directory. In order for this to be accomplished, the file must be assigned with valid keys, if keys exist in the master file directory.

E – Sets the first file header in tape label of the current reel back to skeleton format to logically set the reel to a blank tape.

I – Inhibit cataloguing of a file that was assigned using @ASG,C or @ASG,U.

R – Do a normal @FREE, but do not release any @USE-attached names (unless the A option is also used). Save all pertinent information to reestablish the assignment.

S – Do a normal @FREE of the file but retain the physical tape unit.

X – Remove this run's exclusive-use lock from an assigned file. Take no other action unless indicated by other options.

The action on a @FREE with no options specified is to release all @USE-attached names, and drop filename from the inventory of facilities assigned to the run.

See facilities status bits in Section 4 for definitions of possible rejection or warning bit codes.

@HDG[,options] [heading-text-not-exceeding-96-characters]
@@

Specify a new heading to be printed at the top of each page of printed output, along with the print file's cumulative page number, and the current date.

Options:

N – Suppress printing of the heading, date and page number.

P – Begin page count at 1, instead of at the print file's current page number.

X – Do not print data or page number.

@JUMP label-or-nonzero-number

Transfer control downward to a specified control statement in the run stream, bypassing any intervening control statements or data.

The jump-to control statement is identified either by a label, or by a number which identifies it as the *nth* control statement following the @JUMP (excluding control statements which do not permit a label to be attached).

@LIST[,options] eltname,...

Call the LIST processor to produce an edited dump of one or more absolute, relocatable, or source language elements.

Options:

A,R,or S – Identifies each element as absolute (A), relocatable (R), or source language (S). If A, R, or S is not given, S is assumed.

O – Produce a straight octal dump, without editing.

@LOG message-not-exceeding-132-characters
@@

Enter a message in the master run log.

The semicolon (;) may be used only in its normal capacity, which is to signify card image continuation. The sequence space-period-space (.) also has its usual meaning of start of comment field. The message starts in the second column following the control statement name.

@MAP[,options] SI-eltname,absolute-eltname[,SO-eltname]

Call the MAP processor (the collector) to collect a specified set of relocatable elements, and produce from this an executable program which is in absolute element format.

Options include the standard processor options defined at the beginning of this section and also:

B – The area of main storage occupied by this program need not be zero-filled by the loader prior to loading. Segments specified for indirect loading will not have their main storage areas cleared prior to loading. Any area acquired by MCORE$ will not be cleared.

D – Debug option. Provide diagnostic messages for what appears to be either over 177777_8 (65K) address fields, or errors in instruction formats.

E – Inhibit the possible downward adjustment of D bank starting address, which is normally done automatically when the program's last D bank address exceeds 177777_8 (65K).

F – Mark the absolute output element as being quarter-word sensitive.

R – Merge all of the input relocatable elements into a single output relocatable element, instead of producing an absolute element. External references to elements in the system relocatable library are retained as external references, unless the following source language statement is used: LIB SYS$*RLIB$.

T – Mark the absolute output element as being third-word sensitive.

V – Assign all addresses for both the I and D banks, but then strip off the D bank code. Compare with Y option.

X – If error is detected, terminate collection and exit ERR$.

Y – Assign all addresses for both the I and D banks, but then strip off the I bank code.

Z – Do not prepare the diagnostic tables in the output absolute element which are used by @PMD (and other dump editors) to identify portions of the program by segment name, element name, etc.

The method of collection performed is designated as either a "bank-named collection" or "bank-implied collection." In a bank-implied collection, no banks are explicitly named and the collector generates one I-bank of all odd location counters and one D-bank of all even location counters. In a bank-named collection, the user specifically names all banks (which may number from 1 to 250) and can direct the placement of any or all location counters within each bank. Note that certain source language directives can only be used in bank-named collections.

Here, listed alphabetically, are the most commonly used collector source language directives. These may start in any column, and may use comment fields headed by a space-period-space sequence. In those instances where no source statements are used, the collector includes all relocatable elements from the file TPF$ plus any elements in the file SYS$*RLIB$. which are able to satisfy remaining undefined external references.

COR name-of-included-element

This names an element in an absolute collection which is to have one or more instruction or data words changed at @MAP time. Any number of correction parameter cards may follow the COR card. The first parameter of each parameter card identifies a word address (relative to a location counter in the element) that is to be changed:

rel-addr,location-counter-number

The remainder of the parameter card identifies the replacement value. If this is a single-field number, it is assumed to be a full-word replacement.

If this is two fields, each field may contain any one of three half-word replacements:

externally-defined-symbol[±rel-addr-offset]
rel-addr,location-counter-number
number

If the replacement value is seven fields, the first six of these are the values of the instruction fields f, j, a, x, h, i, (leading zeros must be used to indicate that these are octal values), and the seventh field is the same format as the above half-word replacements or:

rel-addr,location-counter-number,name-of-an-included-element

DBANK[,options] bank-name[,relationship-specification]

Used only in bank-named collection to specify the beginning of the source language defining the named bank. The allowable options are:

C – control bank
D – dynamic bank
E – bank prefers to be loaded in extended storage
$E – bank must be loaded in extended storage
M – initially based on main PSR
P – bank prefers to be loaded in primary storage
$P – bank must be loaded in primary storage
R – bank is read-only
U – initially based on utility PSR
V – assign all addresses, but strip off this bank's code
X – common bank (must be used with M or U options only)

The relationship-specification, if present, must be one of the following formats. Name–n may be a bank-name, a numeric value, or a bank-name ± numeric offset.

name–1	The start address of bank-name is the same as that parameter specified by name–1
(name–1)	The start address of bank-name is the next address (multiple of 01000) following name–1 parameter
(name–1,...,name–n)	The start address of bank-name is the next address (multiple of 01000) following highest of name–1,...,name–n banks.

If no relationship is specified, bank-name starts at the next address (multiple of 01000) following the most recently defined DBANK or IBANK (if omitted on an IBANK statement).

DEF externally-defined-symbol,...

Specifies that the named externally defined labels are to have their names and locations saved in a table (in the absolute element) called ENTRY$. In an R-option collection, the DEF directive must be used to specify those externalized labels which are to remain externalized in the merged output relocatable element.

ENT externally-defined-symbol

Defines a program starting location (which must be in the main segment), to which control should be initially transferred.

EQU undefined-symbol/value,...

Equates a value to an undefined symbol. This value may be either a number or an externally-defined-symbol [±rel-addr-offset].

FORM bank-name
or
bank-name*seg-name

Used only in bank-named collections. Allows duplication of a portion of a program structure previously defined within the map without requiring repetition of the source language used to define that structure. Bank-name specifies the bank whose structure is to be duplicated, and seg-name specifies the segment within the bank whose element inclusion structure is to be duplicated.

IBANK[,options] bank-name [,relationship-specification]

see DBANK

IN eltname-or-common-block-name-or-filename[($lc-set)],...

Explicitly names an element or common block for inclusion in the collection, relative to the positioning of all other elements and common blocks. When a whole file is named, all relocatable elements in the file are included. Common blocks named on IN directives do not have filenames attached to them. The ($lc-set) is allowed only in bank-named collections.

LIB filename
or
filename(bank-name/$lc-set),...

Specifies a file as a library to be searched, prior to the search of SYS$*RLIB$. The file will be searched for elements not yet found, that were named on IN without a file name. If the LIB file has had an entry point table previously prepared by @PREP, it will also be searched to satisfy external references.

The second format is allowed only in bank-named collections and specifies that the named location counter set for any implicitly included element from that file is to be placed in the main segment of the named bank.

NOT eltname-or-filename

Causes the named element, or all elements in the named file, to be excluded from the collection, such as in NOT TPF$. (to prevent an automatic search of that file from being done).

RSEG segname

An RSEG is loaded starting at whatever address (within the program limits) which is in A2 on an ER LOAD$. All instructions and data in an RSEG are collected in the I bank.

SEG segname[*] [,relationship-specification]

Specify the beginning of a new segment, in a program that uses segmentation. The format of segname is a 1-12 character alphanumeric name (with the $ and – characters also allowed) which should not duplicate other element or symbol names in the collection. All elements, etc., named on IN directives will be located within the last segment that was named prior to the IN directive. Elements brought into a collection to satisfy undefined symbols will be located as far to the base of the segment tree as is necessary to ensure that it is at a common juncture of all referencing elements. If * follows segname, any jump to an externally defined I bank label within the segment will cause an indirect call on ER LOAD$ if necessary, to first load the segment (via SYS$*RLIB$, element IDL$). Allowable relationships:

SEG segname	segname immediately follows segment named on last preceding SEG directive
SEG segname,name–1	segname starts at same address as name–1 segment
SEG segname,(name–1)	segname starts immediately following name–1 segment
SEG segname,(name–1,name–2,...,name–n)	segname follows highest last address of segments name–1,...,name–n.

$lc-set

Used only in bank-named collections. Specifies a location counter set via either a keyword or explicit naming as follows:

$ALL	include all location counters
$NONE	include no location counters (used to create dummy or skeleton structures)
$ODD	include only odd location counters
$EVEN	include only even location counters
$$n_1,n_2,...,n_m$$	include only those location counters specified
ALLBUT,$$n_1,n_2,...n_m$$	include all location counters except those specified.

The user code in a program can reference the symbols FRSTI$, LASTI$, FRSTD$, or LASTD$, referring respectively to the program's first and last I and D bank addresses. The collector will replace these symbols with the actual address values assigned. References may also be made to FIRST$, LAST$, and BDI$. These will be satisfied by the first bank address, last bank address and bank descriptor index of the bank in which the reference is made.

@MARK[,option] filename,...

Call the FURPUR processor to write two end-of-file (EOF) marks at the current tape position, and then backspace over the second EOF.

Option:

C – Continue. Do not error terminate, even if errors are detected.

@MODE[,options] filename[,noise/MSA-trans/unit-trans/format]

Set revised mode options and/or noise constant for a tape file.

Mode Options:

B,E,H,I,L,M,O, and V as described in magnetic tape @ASG.

Only those options specifically stated are changed. This is roughly equivalent to the action on an ER IO$ function (42_8), which means that the file must be currently assigned to the run, and even if the file is catalogued, those revised mode settings are not saved in the master file directory.

See facilities status bits in Section 4 for definitions of possible rejection or warning bit codes.

@MOVE[,options] filename,number-of-EOF-marks

Call the FURPUR processor to move a magnetic tape file over a specified number of end-of-file (EOF) marks. The tape is left positioned immediately following the last EOF mark encountered on a move forward, or immediately precedes the last EOF mark encountered on a move backward.

Options:

B – Move tape backward, instead of forward.

C – Do not error terminate, even if errors are detected.

The data which may precede each EOF mark on a tape file is here referred to by the name "file", in double quotes. The use of @MOVE for positioning a tape from one "file" to another is described briefly by the following two examples.

To move forward to the start of a "file" that is three "files" ahead of the current one at which the tape is positioned, @MOVE filename,3 is used.

To move backward, to the start of a "file" that is three "files" behind the current one, requires the following two-command sequence: @MOVE,B filename,4 and @MOVE filename,1.

@MSG[,options] message-not-exceeding-50-characters

Transmit a message to the console operator.

Options:

C,I,H or S – Direct message to either communications console (C), I/O console (I), hardware confidence console (H), or system console (S). If C,I,H,S, (or N) is not given, S is assumed.

N – Include this control statement in the run's printer listing, but otherwise ignore it. @MSG,N functions like a comment card in the run stream; using any other options in addition to N is meaningless.

W – Hold the run in wait status until a response is typed in by the operator. The response will be printed at a demand terminal.

@NUALG[,options] SI-eltname,RO-eltname[,SO-eltname]

Call the NU ALGOL language processor.

Options:

B – List serial number and level number at the beginning of each block in the program during compilation.

E – Required for compiling an external procedure.

F – The compiled code is listed and punched into cards which are accepted by the assembler.

O,R – Remove subscript checking to decrease execution time of production programs.

T – Print the times for the four passes of the compiler.

V – Suppress warning messages.

Y – Suppress warning "This Variable Has Not Been Assigned a Value"

Z – No run-time diagnostic information is prepared.

@PACK[,options] filename,...

Call the FURPUR processor to pack together the nondeleted elements of a program file, by rewriting the file and eliminating the deleted elements.

Options:

A – Treat all but absolute elements as deleted.

O – Treat all but omnibus elements as deleted.

R – Treat all but relocatable elements as deleted.

S – Treat all but source elements as deleted.

@PCH,option[s] eltname

Call the FURPUR processor to punch out onto cards a complete copy of a program file element, preceded by a properly formatted @ELT,I card.

Options include A,O,R,S and C, as defined under @FIND. These may be used in any combination. And also:

H – Used with S option causes sequence numbers to be punched in columns 73-80.

The @ELT,I card that is prepared by @PCH has the same eltname as is specified above. (For a proc element, @PDP,I is punched instead of @ELT,I.)

@PDP[,options] SI-eltname,SO-eltname

Call the procedure definition processor to produce program file proc entries suitable for use by the @ASM, @COB, or @FOR processors.

Options include A,I,L, and X, earlier described under standard processor options, and also:

C – COBOL proc element. Compare with F option.

F – FORTRAN proc element. If neither C nor F option is used, assembler proc is assumed.

@PMD[,options] name,rel-addr/location-counter,word-count,format]
or [eltname/bankname,rel-addr/location counter,word-count,format]
or (epname/bankname,word-count,format]
or [part–1,part–2,...,part–n]

Call the post mortem dump processor to dump all (or specified portions of) the segments of a program that reside in main storage at the time of program termination. Also dump the run's program control table (PCT), if requested.

Options:

A – Dumps both the I-Bank and the D-Bank portions (if a parameter is used, it must be in the namelist following).

C – Dump only those words that have been changed during program execution.

D – Dump D-bank portions. (If a parameter is used, it must be a namelist, as noted below).

E – Produce no dumps unless program terminates in error.

I – Dump I-bank portions. (If a parameter is used, it must be a name-list, as noted below).

L – Dump elements that were included in the program from the system relocatable library file, SYS$*RLIB$. (Since they do not contain user code, these RLIB$. elements are not normally included in @PMD dumps.)

P – Dump this run's PCT, and also any @MAP-produced tables (such as the segment load table, which is used by ER LOAD$) that may reside at the start of the terminating program's D-bank, ahead of its main segment. The PCT consists of one or more 1000_8-word blocks. These tables are dumped in octal format.

X – Used only in conjunction with A, I or D options, to identify a name-list as an exception list, rather than an inclusion list.

part:

eltname	eltname/bankname
segname	segname/bankname
bankname	eltname/segname/bankname
eltname/segname	

NOTE:

For programs collected with Level 23 and earlier levels of the collector, only the first two names may be used.

The A, D, I, and X options are used with an optional parameter which is different from that shown above, and consists of a namelist of one or more 1-12 character names of segments, elements, or common blocks that were included in the @MAP of the program. When a segment is named, all of its elements and common blocks are dumped. Blank common may be explicitly referenced by its @MAP-given name, BLANK$COMMON. Whether or not the A, D, I, or X options are used, all parameters may be omitted. This means all active segments will be included.

The parameters shown above may reference a single 1-12 character element, common block, segment, or bank. Rel-addr is a dump starting address (0 is assumed, if this is not given), relative either to element location counter 0 (or any other location-counter that is stated here), or relative to the start of the block, in the case of a common block.

Word-count is assumed to be the whole location counter or common block, if not stated.

See explanation of 'format' under Dynamic Dump routines in Section 3. (User-defined formats may be given here within parentheses.) All output is single spaced, regardless of what 'format' spacing is specified. 'D', 'S' and user-defined formats are not applicable for C-option dumps.

Several @PMD control statements may be used in succession, in order to separate the dumps of the terminating program into different parts.

The only control statements which may intervene between the @XQT that calls the program to be dumped, and the @PMD are: @JUMP, @SETC, @TEST, @ADD and @EOF. See comments pertaining to @RUN options N and Y.

@PREP filename-1,filename-2,...,filename-n

Prepare an entry point table for a program file, for use by the @MAP processor in searching a LIB specified program file to satisfy undefined symbols.

@PRT[,options] [name,...]

Call the FURPUR processor to produce an edited listing of:

(1) the entire master file directory; or

(2) the master file directory entries for the specified file, account, or project; or

(3) all catalogued files residing on a specified removable disc pack; or

(4) the table of contents of a program file; or

(5) the images of a source language element; or

(6) all files currently assigned to a run.

With no options or specifications, @PRT will list public files and private files with the same project-id as this run, ordered by projects. If an eltname is given, the S option is assumed.

Options:

C – Continue. Do not error terminate, even if errors are detected.

D – Display the names of all catalogued files currently residing wholly or partially on the named removable disc packs. Each name entry must have the format: pack-id/equipment code (e.g., PACK1/F24).

F – List the master file directory entry of only those files whose filenames are given in the operand field.

I – Display the names of all catalogued and temporary files currently assigned to the run.

L – Used with @PRT,T in demand mode to get a long rather than a short listing from demand terminals.

N – List the master file directory entry for that set of files whose account number is the same as this run's.

P – List the master file directory entries for that set of files whose project-id is the same as this run's.

S – Used with an operand field of one or more source language eltnames to list symbolic elements.

T – List the table of contents (TOC) of the program files whose filenames are given in the operand field. In demand mode, a condensed TOC is printed unless the L option is also specified.

U – Display the current usage of the removable disc pack specified. The format is the same as for D.

@QUAL qualifier
@@

Define a qualifier name to be used with any filename references which have no qualifier, but precede the filename with an asterisk.

@REWIND[,options] filename,...

Call the FURPUR processor to rewind a magnetic tape file to the loadpoint of its first reel.

Options:

C – Continue. Don't error terminate, even if errors are detected.

I – Rewind with interlock.

@RSPAR[,option] filename.element
@@

Restart a program at some previously taken checkpoint (by @CKPAR).

Option:

P – Display error messages on the operator's console as well as in the master log.

filename:

Specifies the name of the assigned program file containing the checkpoint.

element:

Specifies the name of the element which contains the checkpoint.

The error status codes returned by @RSPAR are displayed in the section concerning @RSTRT.

@RSTRT[,[p] [/opt]] run-id,[account] ,filename,checkpoint-number[,reel-number]

Restart a run whose checkpoint dump was saved in the catalogued file filename by @CKPT.

The normal options parameter is divided into two subfields, separated by a slash. The first subfield specifies priority from A to Z of the restarted run, and the second subfield contains the options P or T. (See @CKPT.)

Error Codes (Octal)	Description
01	Unrecoverable magnetic tape error. (2)
02	Unrecoverable FASTRAND-formatted mass storage error. (1,2)
03	Full cneckpoint not found on specified file or partial checkpoint not found on specified element. (1,2)
04	File specified on restart request is on illegal device type. (2)
05	The checkpoint file was not catalogued. (2)
06	A facility for the program to be restarted cannot be assigned. In the case where insufficient units are available for tape assignments, this error is not encountered since Restart will wait until the required units are available. (2)
07	Common bank used by checkpointed run no longer exists. (1,2)
010	The checkpoint was not on the reel that was specified on the restart request. (2)
011	The checkpoint is in error and is incomplete. (2)
012	The next part of the checkpoint is not on the reel that was mounted. (2)
013	Format error. (2)
014	Restart of this run is already in progress. (1,2)
015	Run requesting restart of a partial checkpoint has ESI activities or has write enabled common banks attached. (1)
016	File specified on restart request was not assigned to run requesting restart or element named on restart request was not found on specified file. (1)
017	Transparent request on run having no activities. (1)

NOTES:

(1) May occur during partial restart.

(2) May occur during complete restart.

@RUN[,[p] [/opt] [run-id,account,project-id,run-time/deadline,pages/cards,start-time]

Schedule a new run for initiation, and provide necessary accounting information.

The normal options parameter is divided into two subfields, separated by a slash. The first subfield specifies priority from A to Z of the run, and the second subfield contains regular option letters. A label is not permitted.

All parameters in the operand field are optional, with the exception that if deadline is given, the run-time must also be given.

Options:

B – Treat the run input from a demand terminal as batch input and schedule the run as a batch run.

C – Terminate the run if the number of cards punched exceeds the cards estimate.

E-L – Specifies, in 1000_8 word blocks, the initial size of the run's program control table. E=two main storage blocks, F=three,...,L=nine main storage blocks.

N – Inhibit all postmortem and dynamic diagnostic dumping.

P – Terminate the run if the number of pages printed exceeds the pages estimate.

R – Restart run in the event of a recoverable system failure.

S – Process this run in sequence with the previous run submitted from the same peripheral device. This run will be held until the previous run has terminated.

T – Terminate the run if the SUP usage estimate is exceeded.

Y – Allow postmortem and dynamic diagnostic dumping of processors and programs in the system's absolute library file SYS$*LIB$.

The run-id cannot exceed six alphanumeric characters. RUN000 is used if no other run-id is given.

The account parameter, which is used for billing purposes, cannot exceed 12 alphanumeric characters (with the hyphen – character also allowed). If this is not given, 000000 is used.

Project-id cannot exceed 12 alphanumeric characters (with the $ and – characters also allowed). If project-id is not given, QQQ$ is used.

The pages and cards are estimated output numbers.

Deadline-time specifies when a run must be finished, and start-time specifies the earliest time the run can be considered for execution. Both parameters are specified in terms of elapsed time with run submission time normally used as initial time. If both start-time and deadline-time are specified, start-time becomes the initial time.

Both parameters have the format hhmm (hours,minutes). Prefixing the parameter with a D (Dhhmm) changes the meaning from elapsed time to time of day.

The parameter run-time specifies an estimated SUP time in minutes. An 'S' prefixing the time specifies SUP time in seconds. Note, the deadline time is not honored if no run-time is specified. The format of run-time is

mmmm

where mmmm is minutes.

@SETC[,options] value[/j-designator-mnemonic]

Store a value into T2 of the run condition word.

Options:

A – Clears bit 30 of the condition word allowing a normal ERR$ termination.

I – Sets bit 30 of the condition word inhibiting run termination after a program error terminates. Normal processing continues.

Allowable j-designator mnemonics are T2, S4, or S3. T2 is assumed if none is specified.

@START[,[p] [/opt]] name,set
name,set,run-id,acct-id,project-id,run-time/deadline,pages/cards, start-time

Initiate a run whose control stream is contained in an SDF format source language element or a data file.

Two formats are provided for the @START control statement. Format 1 is used when all parameters from a prestored @RUN control statement are to be used. Format 2 is used when changing all or part of a @RUN control statement.

The filename, or filename portion of eltname, specifies a currently catalogued file.

All portions of the @START control statement, excepting the first two parameters, exactly duplicate the @RUN control statement, and are available simply for overriding priority, options, run-id, account, project-id, run-time, etc., which appear in the @RUN image at the beginning of the SDF format eltname-or-filename.

A substitution is always made to replace the account number of the prestored @RUN control statement. The account number is taken from the @START control statement, if present, or otherwise the requesting run's account number is used.

The set-value is an octal number to be set into the started run's condition word, which can be tested from within the run to determine which actions it is to take (see @TEST and ER COND$).

The end of the started element or file is treated as an implied @FIN of the started run.

Any run initiated via @START from a demand terminal is scheduled as a batch run, with its printed output going to the primary onsite printer.

@SYM[,options] filename,copies,device,part-name-1/part-name-2/part-name-3/.../part-name-n
@@

Direct the queuing of previously-created symbiont files to a specified device, or group of devices, for printing or punching. Also direct currently active primary output file (PRINT$/PUNCH$) to an alternate device.

Options:

A – Specifies all files on the tape are to be printed (punched) in the order they appear on the tape. The filename specified must be a tape file.

C – Used with a punch file when a remote site name is given in the device-type-mnemonic parameter.

D – When specified with the generic name PRINT$ for filename, the current part of the PRINT$ file is deleted at the time the file is closed and no output is produced.

U – Inhibit @SYM's normal action of decataloguing (deleting) filename after processing it for output.

If the filename is a user-defined file, it must be catalogued public and not currently assigned. If an internal-filename is specified, the associated file must be assigned to the run. Otherwise, filename must be a generic name (PRINT$ or PUNCH$).

1 to 63 copies may be specified. If omitted one will be assumed.

Device may identify a specific onsite device, a specific remote site, or a group of onsite devices (that is, the group might be all onsite punches, or all onsite 1004 printers). Device group and remote site identifiers are defined at system generation. If omitted, the devices associated with the run initiation device are assumed.

Part-names specify the labels of the symbiont file parts of a multifile tape to be printed or punched. If omitted, only the first part on the tape is processed. This parameter is not applicable to mass storage files.

@TEST function-mnemonic/value[/j-designator-mnemonic],...

Skip the control statement which follows this statement in the run stream if the j-designated portion of the run condition word has a specified test relationship to a specified value.

Allowable function mnemonics are TE, TNE, TG, or TLE.

Value specifies a positive, octal value not exceeding 12 digits to be compared with that portion of the condition word specified by the j parameter.

Any of the partial-word j-designator-mnemonics may be used, or W to indicate whole word. T2 is assumed if none is specified.

If there is more than one parameter, and the test relationship of any of the parameters is true, a skip occurs.

@USE 1-to-12-character-name,filename
@@

Equate a unique, 1-12 character internal name to a filename, where filename is either another previous-@USE-defined internal name, or a full filename specification.

The purpose of @USE is to resolve possible ambiguity among a set of files in the system which might differ only by qualifier or F-cycle, or to direct references in a control stream to a different filename than that which is stated.

It is not necessary that filename be assigned. On a @FREE of a file, its @USE-attached names are normally discarded (see @FREE). See facilities status bits in Section 4 for definitions of possible rejection or warning bit codes.

@XQT[,options] eltname

Initiate the execution of a program which is in absolute element format.

Options:

Any set of option letters may be used. The user program may recover these through ER OPT$. (See Section 2.) The value is also in register A5 when the program initially gains control.

If no eltname is specified, the most recent absolute element inserted into the run's temporary program file (TPF$) is executed. If there are no absolute elements in TPF$, an automatic @MAP is forced of all the relocatable elements in TPF$ to create an absolute element, which is then executed.

The following registers contain the special values described, when the program initially receives control:

A4 – Program type, which is 4 for demand, 5 for deadline batch, and 6 for regular batch.

A5 – @XQT (or processor call statement) options. This is the same information as is furnished in A0 by ER OPT$.

R1 – Data in Fieldata, in the same format as A0 has following a call on ER DATE$. (See Section 2.)

R2 – Time and Date, in the same format as A0 has following a call on ER TDATE$. (See Section 2.)

R3 – Total accumulated SUPs in units of 200 microseconds.

2. EXECUTIVE REQUESTS (ERs)

ER FUNCTIONS AND FORMATS

In the packet formats, parameters in regular type indicate information that must be supplied by the programmer; parameters in italics indicate information that the executive returns in the packet. Brackets [ ] are used to indicate optional fields.

Filename, when shown as a two-word field, is used to indicate an internal 12-character filename (left-justified and space filled).

This section does not contain the communication or special purpose ERs. *UNIVAC 1100 Series Operating System Programmer Reference, UP-4144* (current version), contains the necessary information.

ABORT$ (12_8)

ER ABORT$

Unconditionally terminate all activities, and then the run (if it is not a demand run). Do not provide register dumps or allow a postmortem dump.

If contingency type 7 has been previously registered by an ER IALL$ and an ER ABORT$ is used to terminate all activities, a single new activity of the same type as the original program is created by the EXEC and it is given control with the full set of registers (contents not saved) at the program's contingency routine address.

ACLIST$ (141_8)

L A0,list-designator
ER ACLIST$

Allow the user to define his own set of ASCII control statements and register them with the executive. The ACLIST$ request is similar to the CLIST$ request. The list may contain a maximum of 62 six-character alphanumeric ASCII control statement names. Analogous to the CLIST$ one-word terminators +0 and −0 are the ACLIST$ two-word terminators: +010010010010010010010010 and +013713713713713713713713713, respectively.

ACSF$ (140_8)

L A0,(image-length,image-addr)
ER ACSF$

Submits ASCII control statements for interpretation and processing during program execution rather than from the run stream.

The interpretation of parameters is identical to that for the CSF$ request. Maximum allowed value for the image-length is 60_{10} words; 21_{10} is assumed if 0 is given.

ACT$ (147_8)

L A0,activity-name
ER ACT$

Activate an activity that previously named itself through an ER NAME$.

Activity-name is the one-word name returned on a NAME$ request. The activity making this request to activate another named activity does not itself have to be named.

APCHCA$ (77_8)

ASCII punch control alternate. (See PCHCA$.)

APCHCN$ (75_8)

ASCII punch control. (See PCHCN$.)

APNCHA$ (73_8)

ASCII punch alternate. (See PNCHA$.)

APRINT$ (70_8)

ASCII print. (See PRINT$.)

APRNTA$ (71_8)

ASCII print alternate. (See PRNTA$.)

APRTCA$
(76_8)

ASCII print control alternate. (See PRTCA$.)

APRTCN$
(74_8)

ASCII print control. (See PRTCN$.)

APUNCH$
(72_8)

ASCII punch. (See PUNCH$.)

AREAD$
(166_8)

ASCII read. (See READ$.)

AREADA$
(167_8)

ASCII read alternate. (See READA$.)

ATREAD$
(170_8)

ASCII print and read. (See TREAD$.)

AWAIT$	**L**	**A0,(activity-id-mask)**
(134_8)	**ER**	**AWAIT$**

Delay further execution of the requesting activity until all specified activities have terminated.

Bits 1 through 35 of the activity-id mask correspond to activity id's 1 through 35 which were created via the FORK$ request. Bit 0 of the activity-id-mask is not used.

AWAIT$ may not be used by the initial activity, since the activity which makes this request, and all of the activities referenced by this request, must have activity id's.

BANK$	**L**	**A0,(length,address)**
(160_8)	**ER**	**BANK$**

Retrieves BDI and pertinent bit flags.

The length parameter when added to the initial address parameter defines the highest relative address of the area for which the BDI is to be returned. If a zero length is specified, a value of one is assumed.

A status is returned in A0 in the following format:

Bits 0–11	The BDI (Bank Descriptor Index)
Bits 12–14	Zero
Bit 15	1 if dynamic bank
Bit 16	1 if bank is used as D-Bank
Bit 17	1 if common bank
Bit 18	1 if defined as a D-Bank
Bit 19	1 if bank is under utility PSR base (1110 only)
Bit 20	1 if bank is write protected

If A0 = 0, the address range passed was wholly or partially outside the user's window.

```
LXI,U   A0,1*/17
ER      BANK$
```

Where A0 is set negative, the above calling sequence retrieves all currently active BDIs in A0 and A1:

A0:
H2 – D-Bank BDI
H1 – I-Bank BDI

A1 (if 1110 PSRU active):
H2 – D-Bank PSRU BDI
H1 – I-Bank PSRU BDI

For 1108, A1 is cleared to zero.

CEND$** **ER CEND$
(100_8)

Notifies executive of completion of contingency processing.

CLIST$ **L,U A0,pktaddr**
(153_8) **ER CLIST$**

Changes the operating mode of READ$ to allow reading control statement images, whose names are specified in the CLIST$ packet.

The packet is a list of from 1 to 62 one-word alphanumeric (left-justified and space filled) control statement names, followed by a one-word list terminator of plus or minus zero.

If the list terminator is plus zero, CLIST$ mode terminates when a control statement not in the list, other than @ADD or @EOF, is encountered.

If the list terminator is minus zero (777777777777_8) all nontransparent control statements that are not in the list and which are not an @ADD or @EOF control statement are bypassed until an @ENDX or @FIN control statement is encountered which turns off CLIST$ mode.

Each name in the list is assigned an index value which corresponds to its position, beginning with the number 1 for the first name. When a name in the list is encountered in the runstream, its index is returned in bits 23–18 of A0 by READ$ or AREAD$. A special index value of 63 (77_8) indicates an @ENDX control statement.

COM$ **L,U A0,pktaddr**
(10_8) **ER COM$**

These two instructions can be generated by the proc:

C$OM pktaddr

Transmits an output message of up to 50 characters to the console display device. If an input character count is specified, control is not returned until an answer is received from the operator.

Packet:

WORD	S1	S2	S3	H2
0	*error-code*	console-class	control bits	*actual-input-char-count*
1		output-char-count (max. 50)		output-buffer-addr
2	expected-input-char-count (max. 50)			input-buffer-addr

Console Class:

0	–	System console
1	–	I/O activity console
2	–	Communications console
3	–	Hardware confidence console
4–7	–	Four additional message categories may also be used to direct messages to specific console devices. The class code for these categories may be defined for individual site applications.

Control bits:
Bit 20=1 indicates that the console message and response are in quarter-word ASCII. Bit 20=0, Fieldata format is assumed.

Actual-input-char-count contains the number of input characters received.

The error-code field contains a COM$ error code defined under contingency type 5 (see Section 4).

The above packet can be generated by the proc call:

C$OMPK[,console-class] output-char-count,output-buffer-addr[,expected-input-char-count,input-buffer-addr]

COND$** **ER COND$
(66_8)

Places the run condition word in A0.

The format of the condition word returned in A0 is:

T1	T2	T3
error-condition-bits	*0 or value-set-by-@SETC*	*0 or value-set-by-SETC$*

Bit 30 – Inhibit ERR mode on ERR$ terminations (set by @SETC,I and cleared by @SETC,A)
26 – Last termination was an ABORT$ (not EABT$)
25 – Last termination was an ERR$
24 – This run has had ERR$ terminations

CRTN$** **ER CRTN$
(35_8)

Notifies the executive that contingency processing is complete. Control is returned at the address (not address +1) specified in H2 of word 0 of the contingency packet.

CSF$ **L A0,(image-word-length,image-addr)**
(17_8) **ER CSF$**

Submits a control statement image for interpretation and processing.

The image must be in the identical Fieldata format, including the character @ in the first sixth of word 0, as it would have been if it had been submitted as a regular control statement in the input run stream.

The image is terminated when a comment terminator of space-period-space is encountered or a space following the last allowable parameter field is encountered, or the image-word-length in the upper half of A0 has been exceeded.

Maximum allowed value for image-word-length is 40 words; 14 is assumed if 0 is given.

The 15 control statements which may be processed via CSF$ are:

@ADD	@CKPT	@RSPAR
@ASG	@FREE	@RSTRT
@BRKPT	@LOG	@START
@CAT	@MODE	@SYM
@CKPAR	@QUAL	@USE

Control statement option letters, when used within CSF$ images, retain their normal definitions.

When certain control statements are submitted, A0 is returned containing status or error information. For facility statements (@ASG, @CAT, @FREE, @MODE, or @USE), when one or more bits of A0 are set it indicates either that the request was rejected, or that it was accepted with one or more precautionary warnings. See Section 4 for facility status bit description.

For a @START request, A0 contains codes as follows:

0 – Request processed normally.
1 – Request rejected due to improper run stream in file.
2 – Request rejected due to file unobtainable.
3 – Request rejected due to element unobtainable.
4 – Request rejected due to filename not specified.

For a @CKPT or @CKPAR request, A0 contains:

H1	H2
checkpoint-number	*checkpoint-status-code*

See Section 1 for checkpoint status codes.

CTS$ The calling sequence must be generated by the proc:
(123_8) **C$TS tscell**

Clears a TSQ lock and, if activities are queued, activates the next activity.

The parameter tscell is the address of a TS cell generated by T$CELL. C$TS clears the TS lock and inspects the queue (tscell,H2). If the queue is not empty, an Executive Request is made to remove and activate the highest priority non-C$TSQ activity from the queue. No Executive Request is executed if the queue is empty.

C$TS accommodates normal one-level indexing (e.g., C$TS tscell,$X_X$). Indirect addressing, index incrementation, use of the Execute instruction (EX), and instruction modification are not supported.

C$TS operates regardless of TSQ registration.

CTSA$ The calling sequence must be generated by the proc:
(124_8) **C$TSA tscell**

Simultaneously clears a TS lock and removes an activity from a C$TSQ wait.

The parameter tscell is the address of a TS cell generated by T$CELL. C$TSA must be used to reactivate an activity previously queued via C$TSQ.

The C$TSA request clears the TS lock for tscell, then inspects the TS queue. If the queue is not empty, an Executive Request is executed which searches the queue for the first activity marked as having done a C$TSQ. If such an activity is found, it is eligible for reactivation. If no activities could be found on the queue, C$TSA operation is identical to C$TS.

At most, one C$TSQ activity is enabled for each C$TSA call, and a C$TSA done before an associated C$TSQ is lost.

C$TSA accommodates normal one-level indexing (e.g., C$TSA tscell,$X_X$). Indirect addressing, index incrementation, use of the Execute remote instruction (EX), and instruction modification are not supported.

C$TSA operates regardless of TSQ registration.

CTSQ$ The calling sequence must be generated by the proc:
(122_8) **C$TSQ tscell**

Provides, in conjunction with C$TSA, selective activity synchronization and data protection.

The parameter tscell is the address of a TS cell generated by T$CELL. C$TSQ, when used with C$TSA, provides an alternative to DACT$/ACT$ for synchronization of activities which communicate via a common data area. The C$TSQ request generates an Executive Request which deactivates the calling activity, places it on the TS queue associated with tscell (as if it had failed a TS on tscell), and marks the activity as having done a C$TSQ. The TS lock for tscell is then cleared, and action similar to C$TS is performed to reactivate any activities which may have been queued for failing a TS on tscell while this activity had it set.

C$TSQ accommodates normal one-level indexing (e.g., C$TSQ tscell, X_X). Indirect addressing, index incrementation, use of the Execute remote instruction (EX), and instruction modification are not supported.

C$TSQ operates regardless of TSQ registration.

DACT$** **ER DACT$
(150_8)

Deactivates the calling activity, which previously named itself through an ER NAME$.

If another activity has already executed an ER ACT$ specifying the activity calling DACT$, then no deactivation occurs and control returns immediately.

DATE$** **ER DATE$
(22_8)

Places in A0 and A1 the current date and time in Fieldata decimal numbers.

	T1	T2	T3
A0	*month* (01 – 12)	*day* (01 – 31)	*year* (last 2 digits)
A1	*hours* (00 – 23)	*minutes* (00 – 59)	*seconds* (00 – 59)

EABT$** **ER EABT$
(26_8)

Identical to ER ABORT$ except that a register dump is provided and any @PMD request is processed.

ERR$** **ER ERR$
(40_8)

Error terminates an activity, or, in the case of the last executing activity, error terminates a program.

EXIT$** **ER EXIT$
(11_8)

Terminates an activity normally or, in the case of the last executing activity, terminates a program normally.

EXLNK$** **ER EXLNK$
(173_8)

Exit from a re-entrant processor (REP). Control is returned to the caller of the REP, which may be either another REP or the main program.

FACIL$ **L,U A0,pktaddr**
(114_8) **ER FACIL$**

Obtain in words 2 through 8 of the FACIL$ packet the external filename, the qualifier name, and the facilities assignment information that are associated with a particular internal filename.

WORD	S1
0	internal-filename
1	
2	*external-filename*
3	
4	*qualifier*
5	
6	*equip-type*
7	(equipment type dependent)
8	

Equipment Type:

0_8 – No equipment assigned to filename or no find
1_8 – UNISERVO VIII-C 7-track
2_8 – UNISERVO VI-C, 7-track
3_8 – UNISERVO VIII-C, hardware translate
4_8 – UNISERVO VI-C, hardware translate
5_8 – UNISERVO VIII-C, 9-track
6_8 – UNISERVO VI-C, 9-track
7_8 – UNISERVO VI-C,
10_8 – UNISERVO VI-C, hardware translate
11_8 – UNISERVO 12
12_8 – UNISERVO 16
13_8 – UNISERVO 12, 9-track
14_8 – UNISERVO 16, 9-track
15_8 – UNISERVO 20, 9-track
16_8 – UNISERVO III-A
17_8 – UNISERVO II-A
20_8 – FH-432
21_8 – FH-880
22_8 – FH-1782
23_8 – Word addressable mass storage simulated on 8414 or 8424 disc
24_8 – Word addressable mass storage simulated on 8440 disc
25_8 – Unitized channel storage
30_8 – FASTRAND mass storage, model II or III on 8460
31_8 – Not Used
32_8 – FASTRAND mass storage, simulated on FH-432
33_8 – FASTRAND mass storage, simulated on FH-880
34_8 – FASTRAND mass storage, simulated on FH-1782
35_8 – 8414 disc or 8424
36_8 – 8440 disc
37_8 – FASTRAND-formatted unitized channel storage
40_8 – Card reader and punch
41_8 – Not used
42_8 – 0920/0926 paper tape
43_8 – Not used
44_8 – High-speed printer (751)
45_8 – 758 multiple high-speed printer subsystem
46_8 – Not used
47_8 – 92/9300 subsystem
50_8 – 1004 subsystem
51_8 – Not used
70_8 – CTS
71_8 – WTS
72_8 – CTMC
73_8 – C/SP
77_8 – Nonstandard device

The three classes of facilities for which ER FACIL$ is most commonly used are tape (for the current equipment types, the code range is 1_8 to 17_8), word-addressable drum (20_8 to 27_8), and FASTRAND-format mass storage (30_8 to 37_8). Words 6 through 8 of the FACIL$ packet for these three classes are as follows:

Magnetic tape:

WORD	S1	S2	S3	S4	T3
6	*equip-type*	*file-mode*	*unit-count*	*rel-f-cycle nbr*	*absolute-f-cycle*
7	*@ASG-option-letter-bit-mask*				
8	*total-reel count*	*logical channel*	*noise constant*	*mode-settings*	

File mode settings:

Bit 24 – Word addressable drum
25 – Reading inhibited (can be set by @ASG,W)
26 – Writing inhibited (can be set by @ASG,R)
27 – Needs write key
28 – Needs read key
29 – Exclusively assigned file

Unit-count is the number of tape units (1 or 2) which are physically assigned to this file.

WORD 7

Bit 35 – If set, system has tape label checking in progress
34 – If set, file assigned as temporary file
33 – If set, assigned unit is downed

Mode-settings: All tape units except UNISERVO 12/16/20

Bits 10 and 11: 00_2 – No translation required
01_2 – Software translation
10_2 – Hardware translation

Bits 12 and 13: 00_2 – Low density
10_2 – Medium density
11_2 – High density

Bit 14: 0_2 – Odd parity
1_2 – Even parity

Mode-settings: UNISERVO 12/16 only:
7-track

Bits set:	9–8	Set to mode set for hardware
	10	Not used; must be zero
	11	Control unit translator
	12	Parity (0 – even; 1 – odd)
	13	Data converter (0 – on; 1 – off)
	14	Medium density (if 15 and 14 are 0, it indicates low density)
	15	High density
	16	6-bit packed MSA data transfer format (if 17 and 16 are 0, indicates quarter-word MSA transfer)
	17	8-bit packed MSA data transfer format

Mode-settings: UNISERVO 12/16 only:
9-track

Bits set:	9–8	Same as 7-track
	10	Same as 7-track
	11	Density 10–1600 FPI; 1–800 FPI
	13–12	Not used; must be zero
	15–14	Must be set to mode set for hardware
	17–16	Same as 7-track

Word Addressable Drum:

WORD	S1	S2	S3	S4	T3
6	*equip-type*	*file-mode*	*granularity*	*rel-f-cycle nbr*	*absolute-f-cycle-nbr*
7	*@ASG-option-letter-bit-mask*				
8	*word-length-of-assigned-area*				

File mode bit settings are identical to that for tape. Granularity is 0 for track, nonzero for position (a position is 64 tracks).

FASTRAND-Format Mass Storage:

WORD	S1	S2	S3	S4	T3
6	*equip-type*	*file-mode*	*granularity*	*rel-f-cycle*	*absolute-f-cycle*
7			*@ASG-option-letter-bit-mask*		
8	*initial-granules-reserved-on-@ASG*			*max-granules-specified-on-@ASG*	

File-mode bit settings and granularity are the same as for word addressable drum.

FACIT$ **L,U** **A0,pktaddr**
(143_8) **ER** **FACIT$**

This request is identical to the FACIL$ request except that an additional packet word must be furnished, into which additional facility information for tape or FASTRAND-format mass storage files is placed.

Word 9 for tape:

T1	S3	S4 and S5	S6
subsystem-number	*unit-number*	*alternate-subsystem*	*alt-unit*

Subsystem and unit numbers are for each tape unit assigned.

Word 9 for FASTRAND-format mass storage:

H1	H2
highest-track-written	*highest-granule-assigned*

FITEM$ **LA** **A0,(pkt-length,pktaddr)**
(32_8) **ER** **FITEM$**

Provides a method to obtain a variable amount of information on file or facility assignments.

Packet:

WORD	
0	internal-filename
1	
2	*filename*
3	
4	*qualifier*
5	
6	device dependent
↕	
12	

An internal filename (left justified and space filled) must be placed in the first two words of the information packet.

The remaining words of the packet are filled as a result of the FITEM$ request.

The minimum packet length is nine words; the maximum packet length is dependent upon the equipment type. (For detailed information on the variable portion of the packet, see the *UNIVAC 1100 Series Operating System Programmer Reference, UP-4144* (current version), Section 7.)

Equipment	Length
Word addressable mass storage, arbitrary devices	11
Whole unit mass storage, communications devices	9
FASTRAND mass storage	11
Magnetic tape, removable disc	13

If the pkt-length is 0377777_8, the maximum amount of information allowable for the equipment type is transferred to the packet. If the pkt-length given is less than nine or greater than the maximum for the equipment type, only seven words are transferred to the packet and an error status is returned in register A0 (see following).

Rejection of the FITEM$ request occurs only if the relative packet address specified in the request packet is invalid; that is, the address falls outside the user's bounds, or the span of the FITEM$ packet violates the user's bounds, or the file-name specified was not assigned to the run. If an invalid filename is encountered, the equipment type cell is zeroed and WORD 1 of the filename is cleared.

The status codes (returned in S1 of register A0) applicable to FITEM$ requests are:

1 – The requested packet length exceeded the allowable maximum.

2 – The requested packet length was less than the allowable minimum.

FORK$ **L A0,(parameter-word)**
(13_8) **ER FORK$**

Registers and initiates a concurrent program activity.

Parameter-Word:

S1	S2	S3	H2
RT-priority	activity-id	registers	entry address

RT-priority, which must be zero if the program is not already real time, specifies a real-time switching level priority (2–35) for the new activity.

Activity-id is optional. If given, it is a number from 1 to 35, and must be different for each separate forked activity that is initiated.

Registers specify what set of registers shall be provided for the new activity (these registers will be initially loaded with the contents of the corresponding registers of the calling activity):

0 Minor set (X8–X11, A0–A5, R1–R3)
1 Major set (all X, A and R registers except R0, X0, A15+3 and A15+4)

IALL$ **L A0,packet**
(101_8) **L A1,(extended mask)** (For TYPE above 14_8)
ER IALL$

Notify EXEC to allow user program to capture its own error and other contingency interrupts (only types specified in the IALL$ packet selection-mask).

Packet:

T1	S3	H2
selection-mask	cont-appl	contingency-routine-address

Selection Mask-Bit Settings for Program Contingencies:
(A1 meaningful only if selection mask in A0 is zero)

Contingency Type (Octal)	Bit Set A0	Bit Set A1	Contingency
1	24	0	Illegal Operation
2	25	1	Guard Mode or Undefined Sequence
3	26	2	Floating-Point Overflow
4	27	3	Floating-Point Underflow
5	28	4	Divide Fault
6	29	5	Restart
7	30	6	Abort
10	31	7	Console Keyin
11	32	8	Test and Set (Real Time Only)
12	33	9	Error Mode
13	34	10	Inter-Activity Interrupt
14	35	11	Breakpoint Interrupt (UNIVAC 1110 only)
15	–	12	CAU/Storage or GRS Parity (UNIVAC 1110 only)

Contingency Application:

0 – Entire program
1 – Only the calling activity
2 – Set by real-time program for ESI activities

Contingency types 1 through 5, 14 and 15 result from actual hardware interrupts. Contingency types 6, 7, 11, and 12 result from the executive generated pseudo interrupts.

On the UNIVAC 1108, arithmetic fault (types 3, 4, 5) A-register clearing on standard action is done by examining the a-field of the offending instruction. No clearing occurs if an Execute remote (EX) instruction was used to execute the offending arithmetic instruction.

On the UNIVAC 1110, the value of PSR bit D20 determines the action taken for arithmetic faults.

D20 Value	Standard Action
1	Action is fully compatible with the UNIVAC 1108.
0	Interrupt never occurs. The appropriate arithmetic result registers are cleared and instruction execution proceeds in line. Register clearing is done if the offending instruction occurred via Execute (EX) instruction.

When the user program registers contingency routines to handle one or more of the arithmetic faults (types 3, 4, 5), D20 must equal 1 in order for the interrupt to occur on the 1110.

For UNIVAC 1110 operating systems on the occurrence of a guard mode, illegal operation, breakpoint or undefined sequence interrupt, the Jump History Stack will be captured and saved in a dedicated area of the user's Program Control Table. The user may examine this history via ER PCT$ or LIJ/LDJ to the PCT bank.

Following an interrupt or pseudo-interrupt, control passes to third word of the contingency routine, whose address is specified in the IALL$ packet. The executive stores the following information into the first two words (parameter area) of the contingency routine:

WORD	S1	S2	S3	H2
0	*error-type*	*error-code*	*cont-type*	*reentry-address*
1	not used		*status bits*	*ER packet address or H1 of status word for undefined sequence*

Reentry-address is the address of the offending instruction or in the case of asynchronous contingencies, the address of the last instruction executed before detection of the error or other condition causing the interrupt. Contingency types are as noted above.

Status bits are applicable only for Guard Mode and Undefined Sequence Interrupts on UNIVAC 1110 Systems.

ER packet addr is applicable only for I/O and console error types. It also contains H1 of the status word if an undefined sequence interrupt occurred (UNIVAC 1110 only). It may also be used on non-I/O error mode contingencies to contain information applicable to the error.

Contingency type 10 (IINT) sets the error type field (S1 of word 0) to 1 for onsite II keyin, or 2 for demand @@X C keyin.

The other parameter fields shown above are used only with contingency type 12, and only where relevant. The error types and error codes for contingency type 12 are discussed in Section 4.

Word 1 is used for auxiliary contingency information in some cases.

II$** **ER II$
(27_8)

Provides a means to define the activity which is to accept any unsolicited console input directed to the program.

The activity executing the II$ request is deactivated as for a DACT$ request. However, the activity need not be named. If named, it may be reactivated using an ACT$ request; and if it also has the inter-activity interrupt contingency registered, it may be activated via an INT$ request. An II$ request when an II$ activity has already been defined for the program is not allowed.

Unsolicited console input of up to six characters in Fieldata is stored (left-justified, space filled) in the activity's A0 register, and the activity is activated.

After activation (by either ER ACT$, ER INT$ or console input), the activity is no longer defined as the unsolicited console input activity. The same activity or some other activity must execute another II$ request to redefine the unsolicited console input activity.

The console input activity is also activated by the remote terminal @@X C keyin. Since no input is actually received, register A0 is space filled.

```
INT$            L       A1,parameter
(33₈)           L       A0,activity-name
                ER      INT$
```

Asynchronously interrupts a named activity.

The named activity specified in A0 will be interrupted and given an activity interrupt contingency (type 13_8). The activity interrupt contingency must be registered for the interrupted activity, either as a program or an activity contingency. Otherwise, the request will be ignored. If the activity to be interrupted cannot be found, the caller will be terminated in error with an error type 04 and error code 032. The activity calling INT$ need not be named.

The interrupted activity will be reactivated from an AWAIT$, DACT$ or II$ state if necessary. In this case, the error address in the contingency packet will point one location before the AWAIT$, DACT$ or II$ call, so that recovery may be done, as usual, by returning to the error address +1. In all other cases, except jumps, the reentry address points to the last instruction executed. If the last instruction executed was a jump, the reentry address points to one location before the destination address for the jump.

If the interrupted activity is in a TWAIT$, the time will be allowed to expire. Multiple INT$ requests for the same activity will be queued and processed serially.

The contents of A1 will be placed in word 1 of the contingency packet.

```
IO$             L,U     A0,pktaddr
(1₈)            ER      IO$
```

These two instructions can be generated by the proc:

```
                I$O     pktaddr
```

Requests an operation on an I/O device as specified in the function field of the I/O packet. Control is returned immediately to the executing program, without waiting for completion of the I/O operation.

I/O Packet:

WORD	S1	S2	S3	H2
0	filename			
1				
2	used by executive system		in-act-id	interrupt-activity-addr
3	*status*	function	*AFC*	*final-word-count*
4	G	word-count		buffer-addr
5	0			drum-addr
6	search-sentinel			
7	0			*search-find-drum-addr*

Function	Octal	Mnemonic
Write	10	W$
Write end of file on tape	11	WEF$
Contingency write tape	12	CW$
Skip write tape	13	SW$
Gather write	15	GW$
Acquire mass storage	16	ACQ$
Absolute write	17	ABSW$
Read	20	R$
Read backward	21	RB$
Read and release	22	RR$
Release	23	REL$
Block read drum	24	BRD$
Read and lock	25	RDL$
Unlock	26	UNL$
Absolute read (privileged user and system only)	27	ABR$
Track search all words	30	TSA$
Track search first word	31	TSF$
Position search all words	32	PSA$
Position search first word	33	PSF$
Search drum	34	SD$
Block search drum	35	BSD$
Search read drum	36	SRD$
Block search read drum	37	BSRD$
Rewind	40	REW$
Rewind with interlock	41	REWI$

Function	Octal	Mnemonic
Set mode	42	SM$
Scatter read	43	SCR$
Scatter read backward	44	SCRB$
Absolute read	47	ABSR$
Move forward	50	MF$
Move backward	51	MB$
Forward space file	52	FSF$
Backspace file	53	BSF$

For I/O functions involving no transfer of data, such as write end of file, only packet words 0 through 3 are required. For tape I/O, only words 0 through 4 are required. For drum I/O not involving a search function or absolute drum addressing, only words 0 through 5 are required.

All I/O status codes are defined under type 1 in Section 4. Several I/O packet fields are also described.

Word 4 of the I/O packet is the ISI access control word shown in Section 5. G is usually zero, specifying incrementation to the next word of the buffer following each one-word transfer.

The buffer-addr is for an area in the user program that is at least as large as the value in word-count. In the case of scatter read and gather write functions, this buffer contains a string of access control words; otherwise, it is for the data being transferred.

The drum addresses in words 5 and 7 are relative to word 0 or sector 0 of the user-assigned file name in the packet. If the file is in FASTRAND mass storage format, the address is in sectors; otherwise, it is in words.

The I/O packet for a tape (I$OT) or drum (I$OD) function can be generated by using one of the following proc calls (undesired parameters may be omitted):

I$OT 'filename',function word-count,buffer-addr,G

I$OD 'filename',function word-count,buffer-addr,G drum-addr,search-sentinel

IOARB$ **L,U** **A0,pktaddr**
(21_8) **ER** **IOARB$**

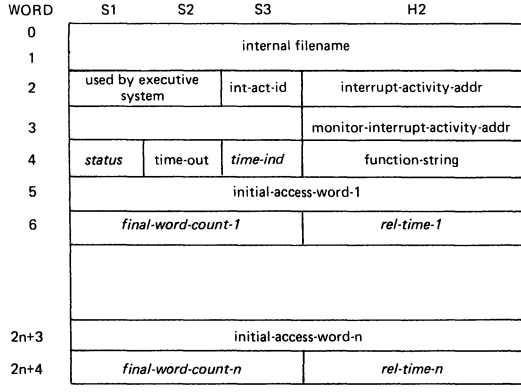
Initiates an arbitrary device I/O operation with control returned, in line, as soon as the request is either listed or the operations have been initiated. An interrupt activity is initiated when the request is completed.

Packet – see IOAXI$.

IOAXI$ **L,U** **A0,pktaddr**
(20_8) **ER** **IOAXI$**

Initiates an arbitrary device I/O operation with the referenced activity simulating an exit function, and controls the return to the program at the appropriate interrupt activity specified in the request packet.

Arbitrary device I/O packet:



The activity performing the IOAXI$ request does not actually exit, but saving and restoring registers is eliminated (except for register A0), and the register set is reduced to the minor set only. The continuation of the IOAXI$ activity at the interrupt point is with the same activity-id; hence, the value in the int-act-id field is ignored for the IOAX$ request.

IOI$ **L,U A0,pktaddr**
(2_8) **ER IOI$**

These two instructions can be generated by the proc:

I$OI pktaddr

Identical to ER IO$ except that when the I/O operation is completed, a specified interrupt activity is initiated at the highest possible priority allowed for this program class.

I/O packet – identical to IO$, except that word 2 contains:

WORD	T1	S3	H2
2	0	int-act-id	interrupt-activity-address

The interrupt-activity-identity (1–35) may be used if synchronization via AWAIT$ is intended with some other activity.

Upon entering the interrupt activity, all registers are destroyed except A0, which contains the I/O packet address. The interrupt activity is limited to using the minor set of registers (X8–X11, A0–A5, R1–R3).

The I$OT and I$OD procs for generating the I/O packet are the same as for IO$, except that two additional subfields are appended to the first parameter field:

'filename',function,interrupt-activity-address,int-act-id

IOW$ **L,U A0,pktaddr**
(3_8) **ER IOW$**

These two instructions can be generated by the proc:

I$OW pktaddr

Identical to ER IO$ except that control is not returned until completion of the I/O operation.

I/O packet – identical to IO$.

IOWI$ **L,U A0,pktaddr**
(24_8) **ER IOWI$**

These two instructions can be generated by the proc:

I$OWI pktaddr

This request combines the features of IOI$ and IOW$. Control is not returned until completion of the I/O operation. Upon completion, a specified interrupt activity is initiated at high priority.

I/O packet – identical to IOI$.

IOXI$ **L,U A0,pktaddr**
(25_8) **ER IOXI$**

These two instructions can be generated by the proc:

I$OXI pktaddr

Identical to IOI$, except that the activity making the request exits.

I/O packet – identical to IOI$.

LABEL$ **L,U A0,pktaddr**
(31_8) **ER LABEL$**

Enable the user to read or write any label block in the first label group on the volume except the VOL1 block.

Packet:

WORD	S1	S2	S3	H2
0	ASCII-Fieldata translation	write-EOF-or label block		label-buffer-addr
1	internal filename			
2				

S1 If set to 0, indicates normal completion; if set to 40_8, indicates abnormal completion (check S2 and S3).

S2 Contains I/O status (if 0, all I/O has completed normally; if a nonzero value, see status for abnormal I/O in Section 4).

S3 If 1, indicates invalid label buffer address.

If 2, indicates that a request was made to read a label following a request to write a label.

If 4, indicates invalid filename.

If 6, indicates an attempt was made to write on a tape file that was not available for writing or an attempt was made to write a label following a request to read a label.

If 10_8, indicates an invalid request (not 1, 2, or 4), or a write EOF request following a read of a label, or a write EOF request before HDR1 has been written.

If 12_8, indicates a request on a labeled tape.

H2 Contains the label buffer address originally supplied by the user.

LCORE$ (44_8)

```
L,U   A0,highest-addr-still-required-in-Ibank-or-Dbank
ER    LCORE$
```

or

```
L     A0,(BDI, highest-addr-still-required-in-bank)
ER    LCORE$
```

Release unneeded main storage at high end of a bank.

The whole bank can be released by specifying the first address of the bank. Before releasing the control bank, the @MAP listing for the program should be checked to ensure that there are no necessary collector produced tables in the control bank.

When an entire segment is in the area of main storage that is released, it is marked as not loaded.

LINK$ (171_8)

```
L     A0,('six-character-repname')
ER    LINK$
```

Transfers control to a specified re-entrant processor (REP).

The specified re-entrant processor must be one of the standard system REPs, or must have been registered with the executive via an RLIST$ request.

Control is transferred to the starting address of the re-entrant processor absolute element.

LOAD$ (111_8)

```
L,U   A0,segname or L A0,(0400000,segname)
L,U   A1,jumpaddr
ER    LOAD$
```

Loads a segment of a program.

If bit 35 of A0 is set, the loader skips the initial clearing of main storage. This causes a faster load, but also causes any reserved areas in the segment that do not initially contain instructions or data to be initially filled with indeterminate contents rather than words of all zeros.

Although the main segment of every program is always automatically loaded at start of execution, and stays loaded throughout execution, it is possible to re-initialize a program by reloading the main segment, using the value 0400000_8 instead of segname. This causes all other program segments to be marked as unloaded.

If jumpaddr is zero, control is returned following ER LOAD$.

If segname was defined at @MAP time as an RSEG, register A2 must also be initialized:

L,U A2,rseg-startaddr or L A2,(bank-name,rseg-startaddr)

The equivalent of the above instructions can be generated by the following proc (undesired trailing subfields may be omitted):

L$OAD segname,jumpaddr,value-of-A0-bit-35,rseg-startaddr,bank-name

MCORE$ (43_8)

```
L,U   A0,highest-expansion-addr-required-for
        Ibank-or-Dbank
ER    MCORE$
```

or

```
L     A0,(BDI,highest-addr-required-for-bank)
ER    MCORE$
```

Obtains additional main storage at high end of I or D bank.

This request cannot be performed while any activities of a program are in real-time status, unless it can be done without moving the program.

ER MCORE$ may be used to create space in an initially void bank.

If the requested address is something less than or equal to the current program size, the request is ignored and control is returned to the user program.

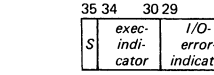
MSCON$ (125_8)

```
L,U   A0,pktaddr
ER    MSCON$
```

Obtains either the entire Master File Directory or entries pertaining to a particular file; provides the means of altering indicators in the directory items.

MSCON$ packets vary according to the function to be performed. For functions and formats of packets, see *UNIVAC 1100 Series Operating System Programmer Reference, UP-4144* (current version).

When MSCON$ returns control to the user, register A0 contains the original packet address in H2, possible error status codes in bits 29–18 and an EXEC-indicator in bits 34–30.



If bit 35 = 0 and bits 29–18 = 0, this signifies normal completion of the requested function.

If bit 35 = 0 and bits 23–18 contain 01, the 01 is a special status code returned by the DREAD$ function signifying that the end of the user buffer has been encountered and there are more directory items to be returned.

If bit 35 = 1 and the error status code is 024, the value in bits 29–24 is the I/O status code received by MSCON$.

If bit 35 = 1 and the error status code is other than 024, the possible status codes are:

020 – Wrong MSCON$ function code in user packet.

021 – User packet not within program limits.

022 – Referenced file is not assigned to this user.

023 – User is referencing a temporary file.

025 – User buffer not within program limits.

026 – User is referencing a nonexistent start item (returned by the DREAD$ function).

027 – User buffer area not large enough (returned by MSALL$ functions). Or user packet specifies zero for number of backup tape reels (returned by the DBACK$ function).

031 – The referenced disc unit has been marked down or reserved (returned by the DGETP$ function).

032 – The user packet specifies an illegal pack-id, or the requested pack-id cannot be found.

033 – The output file initial reserve is too small to contain the current total of system directory items (returned by the DGET$ function).

034 – The cumulative total of system directory items has dynamically expanded beyond the capacity of the output file (returned by the DGET$ function). This situation differs from that described for status code 33_8, in that, in this instance DGET$ has been in process and directory items have been placed on output to the file.

035 – The user program has I/O outstanding, is employing ESI activity, has a count of activities totaling more than one, or is supplying a data buffer which lies in a common bank (returned by functions DGET$ and DGETP$).

036 – The packet specifies a maximum cycle range value not in the range $1–32_{10}$, or less than the current range value (returned by the DCYC$ function).

037 – The user is neither privileged, nor has the subject file been assigned with correct read/write keys (returned by the DBIT$ function).

NAME$ **L,U** **A0,activity-name**
(146_8) **ER** **NAME$**

Attaches a name to the calling activity for purposes of later referencing this activity via an ACT$ or DACT$ request.

The executive expands the 18-bit activity-name supplied in A0 to a full word by inserting three additional characters in the upper portion of A0. This full word name, which the executive returns in A0, must be used with subsequent ACT$ requests. The user supplied name must be unique within a program.

NRT$ **ER** **NRT$**
(62_8)

Returns a real-time activity to original program type.

OPT$ **ER** **OPT$**
(63_8)

Obtains in A0, in master bit notation, the option letters from the @XQT or other control statement that caused this program's execution.

Master bit notation means that bit 0 is set for a Z option, bit 1 is set for a Y option, . . ., and bit 25 is set for an A option.

PCHCA$ **L** **A0,(image-word-length, image addr)**
(165_8) **ER** **PCHCA$**

Specifies control functions for a user-specified punch alternate (PNCHA$) file.

This request is similar to PCHCN$, except that the first two words of the image specified by 'image addr' contain the name of the alternate file to which the functions are to be applied.

The ASCII equivalent of PCHCA$ is APCHCA$ (77_8).

PCHCN$ **L** **A0,(image-word-length, image-addr)**
(164_8) **ER** **PCHCN$**

Specifies control functions for the symbiont PUNCH$ file.

See image definition under PRTCN$. The functions that are applicable to PUNCH$ files are:

S,text-requesting-special-forms.
C,options.

The options relating to the C function, which causes the output symbionts to react as the input symbionts would to a @COL control statement, are:

B – Switch to column binary.
E – Switch to 80-column code.
R – Insert a logical break into punch output files which is used by the punch symbionts when skipping forward through a punch file.
W,line width – Changes the maximum card length from 14 words to the value specified.

The following additional options are recognized by the 9300 symbiont and are ignored by other card punch symbionts:

1100 – Switch to Fieldata punch code pattern
9000 – Switch to EBCDIC punch code pattern
ASC – Switch to ASCII punch code pattern

PCT$
(64_8)

```
L     A0,(word-count-from-1-to-1000₈,buffer-
        addr)
ER    PCT$
```

Obtains a copy of requested portions of this program's program control table (PCT) in user-specified buffer.

Optional calling sequence:

```
L,U   A0,buffer-addr
L     A1,(word-count,relative-starting-addr-within-PCT)
ER    PCT$
```

The first calling sequence assumes a relative starting address within PCT of 0.

The PCT formats are lengthy, technically complex, and subject to occasional changes. Refer to the latest 1100 Series Systems Memorandum and other current documents.

PFD$
(106_8)

```
L,U   A0,pktaddr
ER    PFD$
```

Sets the 'deleted' flag for a program file element.

Packet – see words 0 through 7 of PFI$.

PFI$
(104_8)

```
L,U   A0,pktaddr
ER    PFI$
```

or

```
L     A1,next write location
LN,U  A0,pktaddr
ER    PFI$
```

Inserts new element table entry into a program file table of contents. If this element's name, version, and type duplicate that of another nondeleted element in the file, marks that other element deleted. The second calling sequence combines the functions of PFI$ and PFUWL$.

WORD	T1	S3	H2
0	filename		
1			
2	element name		
3			
4	version-link-sequence-nbr		pointer-link-sequence-nbr
5	flag bits	elt-type	type-link-sequence-nbr
6	element-version name		
7			
8	(type dependent: see following two-word format diagrams)		
9			
10	sector-location-of-text-within-file		
11	time-and-date-of creation		

Flag Bits:
35 – Deleted
30 – Arithmetic fault non-interrupt mode*
29 – Arithmetic fault compatibility mode*
28 – ASCII symbolic
26 – Third-word sensitive
25 – Quarter-word sensitive
24 – Marked in error

Type of Element:
1_8 – Symbolic
2_8 – Assembler procedure
3_8 – COBOL procedure
4_8 – FORTRAN procedure
5_8 – Relocatable
6_8 – Absolute
7_8 – Omnibus

**NOTE:*

The combined settings of bits 29 and 30 are meaningful only with regard to 1110 execution and are interpreted as follows:

00 – Unknown arithmetic fault mode
01 – Arithmetic fault non-interrupt mode
10 – Arithmetic fault compatibility mode
11 – Insensitive arithmetic fault mode

Time-and-date is identical to the word returned by ER TDATE$, except that H1 and H2 are reversed.

Symbolic and Procedure Elements:

WORD	35 31	30 24	23 18	17 12	11 0
8	cycle-limit		latest-cycle		cycle-count
9	processor	0		sector-length-of-text	

Processor:

0_8 – unmarked	5_8 – @ALG	12_8 – @APL
1_8 – @ELT	6_8 – @MAP	13_8 – @BASIC
2_8 – @ASM	7_8 – @DOC	14_8 – @LISP
3_8 – @COB	10_8 – @SECURE	15_8 – @PLUS
4_8 – @FOR	11_8 – @SSG	

Relocatable Elements:

WORD	H1	H2
8	sector-location-of-Preamble-within-file	
9	sector-length-of-Preamble	sector-length-of-text

Absolute Elements:

Level 23 collector

WORD	H1	H2
8	program-I-bank-word-length	program-D-bank-word-length
9	first-D-bank-addr-in-program	sector-length-of-text

or

Post level 23 collector

T1	T2	T3
0400 + other flag bits	no. of user-banks	no. of common banks
0	sector length of text	

PFI$ uses words 2 through 11 to build the standard 10-word element table entry.

The five ERs that begin with PF are called the program file package (PFP). Their operations are similar to several routines in the SYS$*RLIB$ element BSP (basic service package).

When the PFP returns control to the user, a status is returned in A2:

0_8 – Normal status
1_8 – No find (for operation on existing element)
2_8 – I/O error
3_8 – Program file not defined
5_8 – Program file overflow

PFS$
(105_8)

```
L,U    A0,pktaddr
ER     PFS$
```

Search program file table of contents for an element of the name (and version) and type indicated. Fill in the remainder of words 4 through 11 of the packet with complete element table information.

Packet – identical to PFI$, except only name and type must be furnished.

PFUWL$
(107_8)

```
L,U    A0,('filename  ') or L,U  A0,pktaddr
L      A1,(sector-address)
ER     PFUWL$
```

Update program file's next available write location to the sector address given in A1.

Packet – see words 0 and 1 shown for PFI$.

The two-word literal created above, where the filename in quotes has 12 characters (left-justified and space filled), contains the same contents as the first two words of the packet shown for PFI$.

PFWL$ **L,U A0,('filename ') or L,U A0,pktaddr**
(110_8) **ER PFWL$**

Obtains in A1 the program file's next available write location. This is the address of the first sector following the existing text in the file.

See comments for PFUWL$.

PNCHA$ **L,U A0,pktaddr**
(145_8) **ER PNCHA$**

These two instructions can be generated by the proc:

P$NCHA pktaddr

Transfers a Fieldata image to a user-specified punch alternate (PNCHA$) file.

Packet:

WORD	T1	S3	H2
0	not-used	word-count	buffer-addr
1	12-character-Fieldata-filename		
2			

The ASCII equivalent of PNCHA$ is APNCHA$ (73_8).

PRINT$ **PF FORM 12,6,18**
(16_8) **L A0,(PF line-spacing,word-count,buffer-addr)**
ER PRINT$

Assuming the same print FORM definition, these two instructions can be generated by the proc:

P$RINT (PF line-spacing,word-count,buffer-addr)

Transfers a Fieldata image to the symbiont PRINT$ file. If an image is greater than 22 words, the printing of the PRINT$ file is aborted.

The line-spacing field is usually set to 1. Maximum value is 2047. If images to be printed are in ASCII, they may be 33 words long, and an ER APRINT$ (70_8) is used instead of ER PRINT$.

PRNTA$ **L,U A0,pktaddr**
(144_8) **ER PRNTA$**

These two instructions can be generated by the proc:

P$RNTA pktaddr

Transfer a Fieldata image to a user-specified print alternate (PRNTA$) file. If an image is greater than 22 words, the printing of the PRNTA$ file is aborted.

Packet:

WORD	T1	S3	H2
0	line-spacing	word-count	buffer-addr
1	12-character-Fieldata-filename		
2			

The action here is similar to that for PRINT$. The ASCII equivalent of PRNTA$ is APRNTA$ (71_8).

PRTCA$ **L A0,(image-word-length,image-addr)**
(155_8) **ER PRTCA$**

Specify control functions for a user-specified print alternate (PRNTA$) file.

This request is similar to PRTCN$, except that the first two words of the image specified by 'image addr' contain the name of the alternate file which the functions are to be applied to.

The ASCII equivalent of PRTCA$ is APRTCA$ (76_8).

PRTCN$ **L** **A0,(image-word-length,image-addr)**
(137_8) **ER** **PRTCN$**

Specifies control functions for the symbiont PRINT$ file.

Image is a string of Fieldata-coded functions, each of which begins with a function letter, has its parameters separated by commas, and ends with a period. Spaces before, between, or after the functions in the image string are ignored.

The functions that are applicable to PRINT$ files are:

A	–	allow error recovery.
I	–	inhibit error recovery.
L,	–	line-number-to-which-printer-should-be-spaced.
H,	–	option, starting-page-number, text-of-heading.
M,	–	number-of-print-lines, top-margin,bottom-margin.
R	–	insert a logical break into the print output file.
S,	–	text-requesting-special-forms.
W,line width	–	change print line width to value specified.

The options relating to the H function for heading are:

X – Suppress printing page number and date.
N – Do not print heading.

The ASCII equivalent of PRTCN$ is APRTCN$ (74_8).

PSR$ **L** **A0,(bits-as-defined-below)**
(157_8) **ER** **PSR$**

Sets 0 or 1 into any of three processor state register (PSR) bit positions. Obtains former PSR word in register A1 (first word of main PSR for 1110).

Six bits in A0 have defined meanings:

If bit 0 = 1, set PSR quarter-word mode to value of bit 17.
If bit 2 = 1, set PSR double-precision underflow to value of bit 32.
If bit 3 = 1, set PSR floating-point compatibility to value of bit 35.

PUNCH$ **L** **A0,(word count,buffer-addr)**
(130_8) **ER** **PUNCH$**

These two instructions can be generated by the proc:

P$UNCH **word-count,buffer-addr**

Transfers a Fieldata image to the symbiont PUNCH$ file.

The ASCII equivalent of PUNCH$ is APUNCH$ (72_8).

READ$ **L** **A0,(end-of-file-jump-addr,buffer-addr)**
(15_8) **ER** **READ$**

These two instructions can be generated by the proc:

R$EAD **end-of-file-jump-addr,buffer-addr**

Obtains the next Fieldata card image from the input run stream.

If the original input was in ASCII code, READ$ translates it into Fieldata.

If the code is wanted in ASCII instead of Fieldata, ER AREAD$ (166_8) should be used instead of ER READ$, and a buffer of 20 words instead of 14 words must be provided if the run stream is from a card reader. AREAD$ does translation to ASCII, if necessary.

If READ$ encounters an end-of-file (EOF) image instead of a normal data image, control is transferred to the specified end-of-file-jump-addr, with S6 of A0 containing any sentinel character which appeared in column 6 of the @EOF statement.

Upon normal return from READ$, H2 of A0 contains a count of the number of words transferred, which excludes trailing words of all spaces. Other bits may be set in A0 as follows:

Bit 35 – Next control statement in the run stream cannot be passed to the user. Do not attempt additional READ$ requests. (Set only on return to EOF address.)

34 – Currently reading from an @ADDed file or element.

33 – E option on @ADD. (Set only on return to EOF address.)

31 – Image is in the internal format (INFOR) of a control statement.

30 – Used when bit 31 is set, to indicate that there are more INFOR words following those which have thus far been read for this control statement.

29–18 – If nonzero, this is the CLIST$ index value, after encountering a CLIST$-specified control statement. (See ER CLIST$.)

READA$	**L,U**	**A0,pktaddr**
(42_8)	**ER**	**READA$**

These two instructions can be generated by the proc:

R$EADA pktaddr

Obtains a Fieldata card image from a user-specified read alternate (READA$) file.

Packet:

WORD	H1	H2
0	end-of-file-return-addr	buffer-addr
1	12-character-Fieldata-filename	
2		

The action is similar to that for READ$. The ASCII equivalent of ER READA$ is ER AREADA$ (167_8).

The file named in this packet must be previously assigned, and in standard data file (SDF) format.

RLINK$	**L**	**A0,('six-character-repname')**
(172_8)	**ER**	**RLINK$**

This request is the same as ER LINK$ except that when it is used from within one re-entrant processor (REP) to chain out to another REP, the return point saved is not back to the calling REP, but to the return point that the calling REP itself would have used.

RLIST$	**L**	**A0,(entry-count-from-1-to-20,**
(175_8)		**pktaddr)**
	ER	**RLIST$**

Registers a list of nonstandard re-entrant processors (REPs) with the executive.

The packet is a single two-word file name, identifying where the REPs in the list reside, plus as many one-word Fieldata repname entries as are specified in the entry count.

Each RLIST$ request causes previous RLIST$ entries for the calling run to be destroyed. An RLIST$ request with an entry count of zero deletes all entries.

RT$	**L,U**	**A0,switching-priority-level-from-2-to-**
(61_8)		**35**
	or	
	LN,U	**A0,switching-priority-level-from-2-to**
		35
	ER	**RT$**

Upgrades program status to real time, or changes the switching priority of an activity that is already real time.

If contents of A0 are negative, program will not be repositioned.

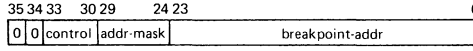
SETBP$	**L,U**	**A1,BDI**
(156_8)	**L**	**A0,(breakpoint parameter)**
	ER	**SETBP$**

Sets the UNIVAC 1110 programmable breakpoint register.

The equivalent calling sequence may be generated via the S$ETBP procedure using the following call:

S$ETBP breakpoint-addr control address-mask BDI

The format of the breakpoint parameter is:



breakpoint-addr — The relative address to be compared to either an instruction or storage operand address. Only one comparison can be made at a time.

addr-mask — Control bits which when set, force equality comparison for corresponding bits 5–0 of the relative breakpoint address. Bit 24 controls bit 0, bit 25 controls bit 1, etc. Any combination of bits can be set. This allows breakpointing on a block of 077 addresses.

control — Bit 30 specifies initiation of a breakpoint interrupt during a store instruction. A 'W' is used to specify this bit if S$ETBP is used.

Bit 31 specifies initiation of a breakpoint interrupt during a read from storage. An 'R' is used to specify this bit if S$ETBP is used. Both bits 30 and 31 ('W', 'R') can be specified simultaneously.

Bit 32 specifies initiation of a breakpoint interrupt for an instruction address comparison. This bit must not be set if either bit 30 ('W') or bit 31 ('R') is set. A 'P' is used to specify this bit if S$ETBP is used.

Bit 33 specifies that a BDI is supplied in A1. If a BDI is supplied on the S$ETBP call, this bit is automatically set.

Bits 34 and 35 are not used, but must be zero.

The BDI supplied in A1 is accepted only when bit 33 is set in the breakpoint parameter and permits setting of a breakpoint to a bank not currently in the program's addressable area. If a BDI is not specified (bit 33 not set), on return, A1 will contain either the previous BDI if a SETBP$ has previously been initiated and was currently active or the BDI reflecting the breakpoint address specified in the breakpoint parameter. If the relative address specified overlaps more than one bank, the BDI is determined from the active PSR.

To clear the breakpoint setting, A0 is set to zero on the ER SETBP$ request.

If bits 30–32 are zero on the ER call, bit 32 (P-bit) is automatically set.

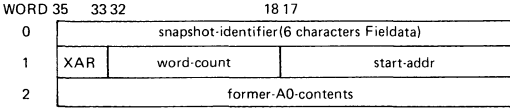
SETC$	**L,U**	**A0,any-number-from-0-to-7777$_8$**
(65$_8$)	**ER**	**SETC$**

Sets the contents of the lower third of A0 into T3 of the run's condition word.

The number in A0 can represent a program status which can be retrieved by later programs in the run using COND$, or subsequently tested from the control stream using @TEST.

SNAP$	**S**	**A0,pktaddr+2**
(126$_8$)	**L,U**	**A0,pktaddr**
	ER	**SNAP$**

Obtains a snapshop dump of selected control registers and areas of main storage.



The XAR field is the three high order bits of word 1, and specifies which sets of control registers to dump:

0_8	None	4_8	Only X
1_8	Only R	5_8	X and R
2_8	Only A	6_8	X and A
3_8	R and A	7_8	All registers

The former-A0-contents are dumped as the value of A0, and restored automatically into A0 following the SNAP$.

The following proc call generates in sequence, the above three instructions, a J $+4 instruction, and the above three-word packet, which is everything needed to accomplish a SNAP$ request:

L$SNAP 'snapshot-identifier',XAR,word-count,start-addr

TDATE$ **ER** **TDATE$**
(54_8)

Obtains in A0 the current date and time in binary code.

A0 Format:

	S1	S2	S3	H2
A0	*mm*	*dd*	*yy*	*time-in-seconds-from-midnight*

mm is month, from 1 to 12
dd is day, from 1 to 31
yy is year, modulo 1964

TFORK$ **L** **A0,(parameter-word)**
(14_8) **L,U** **A1,wait-time-from-2-to-30000-in-milliseconds**
ER **TFORK$**

Registers a concurrent program activity, following the wait time specified in register A1.

This is identical to ER FORK$, except for the wait time specification. 30,000 milliseconds is 30 seconds.

There is no delay in the calling activity.

For a real-time activity, the wait-time specified may exceed 30 seconds.

TIME$ **ER** **TIME$**
(23_8)

Obtains in A0 the current time in milliseconds past midnight.

TINTL$ **L,U** **A0,('filename ')**
(136_8) **ER** **TINTL$**

Reinitializes a tape file containing one or more reels back to load point of the first reel, to allow an additional pass of the file.

The above two-word Fieldata literal is 12 characters, left-justified and space-filled.

TREAD$ **L,U** **A0,pktaddr**
(102_8) **ER** **TREAD$**

This is the equivalent of a PRINT$ request followed by a READ$ request.

Packet:

WORD	T1	S3	H2
0	line-spacing	image-length	output-buffer-addr
1	end-of-file-return-addr		input-buffer-addr

Word 1 of this packet, and register A0 after control is returned, are handled in the same manner as on ER READ$.

If the characters being transmitted are in ASCII rather than Fieldata, ER ATREAD$ (170_8) is used instead of ER TREAD$.

TSQCL$ **ER** **TSQCL$**
(113_8)

Deregisters Test and Set queuing.

Returns Test and Set conflict processing to normal mode. If TSQ mode is cleared with activities still queued, they will remain queued until a CTS, CTSA, or C$TSQ is done.

TSQRG$ **ER** **TSQRG$**
(121_8)

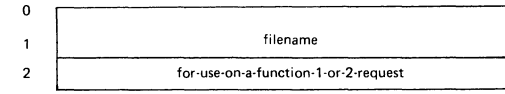
Registers automatic queuing of Test and Set conflicts.

Activities which encounter Test and Set conflicts on T$CELL cells will be de-activated and queued by the executive.

TSWAP$ **L** **A0,(function,pktaddr)**
(135_8) **ER** **TSWAP$**

Close the current reel for a tape file and request loading of the next reel of the file.

Packet:



Inclusion of a function in H1 of register A0 indicates the following:

Function	Description
0	Swaps to the next reel of the file.
1	Swaps to the next reel of the file and places the reel number in word 2 (3rd word) of the request package.
2	A request is made to mount reel specified in word 2 of the request packet. If this reel is not currently recorded as part of the file, it is added as the last reel.

TWAIT$ **L,U** **A1,wait-time-from-2-to-30000-in-milliseconds**
(60_8) **ER** **TWAIT$**

Delays execution for a specified timed wait period.

Note that A1 is used, not A0. 30,000 milliseconds is 30 seconds.

For a real-time activity, the wait time specified may exceed 30 seconds.

T$CELL **tscell** **T$CELL**

This is not an ER, but rather a proc for generating special test-and-set cells for use with Test and Set queuing. tscell is any label by which the cell is referenced.

UNLCK$ **ER** **UNLCK$**
(67_8)

This ER enables an I/O interrupt activity to reduce its switching priority to the priority of the activity which initiated the I/O request.

UNLNK$ **ER** **UNLNK$**
(174_8)

Unlinks a re-entrant processor (REP) by returning control directly back to the main user program, even though there may be other REPs in the calling chain between this REP and the main program.

WAIT$ **TP** **pktaddr+3**
(6_8) **ER** **WAIT$**

Delays execution until the I/O operation controlled by a specified I/O packet has been completed.

I/O packet: (See IO$.)

WANY$ **ER** **WANY$**
(7_8)

Delays execution until any current I/O operation is completed.

An error results if no I/O operations are still in process or if none have completed since the last waiting type ER request.

An LCORE$ request will have caused all outstanding I/O to be completed.

3. LIBRARY SUBROUTINE PACKAGES

These packages consist of sets of one or more relocatable subroutine elements which, if referenced by any of the subroutine entry point names given below, become a part of the user's program when it is @MAPped. These elements reside in the system relocatable library file, SYS$*RLIB$.

The reentrant subroutines may be called on simultaneously by different activities of the same program. The term re-entrant is defined as: that an activity executing the subroutine may be interrupted many times, and then each time re-enter the subroutine for resumption of execution following the point of interruption, without having any of its necessary data destroyed by other activities that were executing the same instructions during the original activity's interruption. Since each activity has its separate set of control registers, and also its separate calling packet and buffers, all changeable data in a re-entrant subroutine must be kept in registers or in one of these user-supplied locations.

BLOCK BUFFERING PACKAGE (BBP)

The block buffering package (BBP) is a set of re-entrant subroutines, which may be used to simplify the reading in or writing out of blocks of data in main storage, either to or from tape or FASTRAND-format mass storage files. While one block of data is being written out, for example, another area of buffer is automatically made available to the user for setting up a further write. Blocks of data in a FASTRAND-format file may be referenced randomly (nonsequentially) by their relative block number, if desired, without regard to actual sector addresses or whether a given block happens to fill only partial sectors.

When block buffering is initialized for a particular file (see: BOPEN$ subroutine), the file is defined as being in input, output, or in/out mode. Restrictions on these modes are:

Input — no writing operations are permitted

Output — no reading operations are permitted

In/out — only for FASTRAND-format mass storage files (both input and output operations are permitted)

For sequential read and writes, block sizes may be either fixed or variable in length.

Random (nonsequential) reads and writes are restricted to FASTRAND-format mass storage files, use a fixed length block size, and require that a block number be specified.

BBP precedes each variable-length block on FASTRAND mass storage with a single word which specifies the block's word size.

On return from each BBP open or read operation for an input or in/out mode file, register A0 contains:

H1	H2
word-count-of-block-read	*addr-of-block-read*

On return from each BBP open, write, mark, or close reel operation for an output or in/out mode file, register A0 contains:

H1	H2
word-count-of-next-block-to-write	*addr-of-next-block-to-write*

The word-count in H1 is either a fixed-length block size or, in the case of a variable-length block file, the maximum area available in the buffer. When an in/out mode file is opened, the block pointed to by A0 is both the first block of the file, that has already been read in, and the block that will be written out on the next write request. This allows the file's label block to be easily updated.

When a cataloged file is closed by BBP, its block size and (if FASTRAND mass storage) end-of-file sector address are saved in the master file directory. The complement of the block size is used if the block size is variable. This information is then automatically retrieved, if not specified, on a subsequent opening of the file.

BBP works with a 25-word file control table (FCT) having the following format, where words 0 through 5 are an I/O packet of the format shown for ER IO$ in Section 2.

WORD	S1	S2	S3	S4	S5	S6
6	open-flag	look-ahead	*file-mode*	*current-buffer-held-by-I/O*		
7	max-block-word-count			fixed-block	*lock-flag*	*FAST-RAND-flag*
8	[I/O-error-exit]			[BBP-call-error-exit]		
9	*user-buffer-starting-addr*			[abnormal-condition-exit]		
10	[sentinel-value]					
11	addr-of-buffer-packet			*activity's-reentry-addr*		
12	*cumulative-block-count*					
13	*addr-of-last-of-queued-buffers*			*addr-of-first-of-queued-buffers*		
14	*current-data-location-in-buffer*			*save-of-BBP-routine-return-addr*		
15	*word-length-of-data-block*			*data-buffer-starting-addr*		
16	*item-flag*	*frame-count*	*CKPT/RSTRT*	0	*I/O-flag*	*queue-count*
17	*exclusive-read-return*			*relative-FCT-addr*		
18	*sector-count*					
19	0	*tape-equip*	*IN/OUT-read*	0	*abnormal-lock*	*activity-switch-location*
20	*highest-FASTRAND-mass-storage-addr*					
21	*FASTRAND-mass-storage-end-of-file-addr*					
22	[test-and-set]	[sentinel-pos]	*read-option*	*user-reentry-address*		
23	[mask-for-block-sentinel-check (all bits set, if unspecified)]					
24	[8-word-register-save-location]					

Most of the fields are initially zero-filled prior to a call on BOPEN$. The fields shown in italics do not require further attention by the user, and are used for internal control purposes by BBP.

Fields which must be specially initialized are:

Filename in words 0 and 1 (see I/O packet under ER IO$).

Open-flag which is set to 1 to show that FCT is unopened.

Fixed-block which is set to 1 if block length is fixed; otherwise zero.

Addr-of-buffer-control-packet (see BPOOL$).

Other fields which may be set are:

Look-ahead – normally should be set to (and may not exceed) the number of buffers in the buffer pool less 1. This is the number of blocks to read ahead for input files.

Max-block-word-count – this is set to the maximum possible value (buffer size minus three words) if unspecified.

I/O-error-exit – address to which control should be transferred (instead of taking direct ERR exit) if an I/O error is encountered. The I/O error code (see type 1 in Section 4) is returned in H1 of A1, and the user's re-entry address to resume processing is returned in H2 of A1.

BBP-call-error-exit – address to which control should be transferred (instead of taking direct ERR exit) if one of the following BBP call error conditions is encountered. The error code is returned in H1 of A1, and a re-entry address in H2 of A1.

BBP Call Error Codes:

1_8 – Missing buffer pool link.

2_8 – Request made to close a previously closed file.

3_8 – Request made to open a previously opened file.

4_8 – Request to read or write a closed file.

5_8 – Request to write a block greater than maximum block size, or rewrite a block in in/out mode and size requested to write is greater than size read.

6_8 – Specified FASTRAND mass storage variable block size exceeds maximum block size or an attempt was made to read variable blocks from a fixed-block file.

7_8 – A random request was made on a file that was not assigned to FASTRAND mass storage.

10_8 – Random request made and block size is not fixed.

11_8 – Insufficient buffers in pool to satisfy look ahead for input or output.

12_8 – Invalid block number for random read request.

13_8 – Read request for a block greater than block size read.

14_8 – File not assigned to FASTRAND mass storage for in/out mode.

15_8 – Random write request for input file.

16_8 – Random read request for output file.

17_8 – Read request with move-length parameter specified but no move-address specified.

20_8 – Read request for output file.

21_8 – Buffer size less than specified block size.

23_8 – Location of link or buffer area outside user's assigned area.

24_8 – Block size not fixed for reverse mode for FASTRAND mass storage file.

25_8 – No I/O facilities assigned or improper equipment type.

26_8 – Write request for input file.

27_8 – Mark request for input file.

64_8 – Invalid mode parameter for open request.

Abnormal-condition-exit – address to which control should be transferred when any of the following conditions are encountered. The condition code is returned in H1 of A1, and a re-entry address in H2 of A1.

Condition Codes:

1_8 – End-of-file or load point encountered for input tape file.

2_8 – End-of-tape encountered for output file.

6_8 – Sentinel block encountered.

10_8 – Block of an in/out mode file that was previously read exclusively has been timed out by the system.

Sentinel-value, Sentinel-position, and Mask-for-block-sentinel-check – a sentinel value not exceeding 36 bits, which is tested against the relative word in each block specified by sentinel position, ignoring those bit positions not set in mask for block sentinel check, to identify a "sentinel block".

8-Word-register-save-location – an address in the user program where registers A1 – A5 and R1 – R3 can be saved by BBP. If not specified, these registers are saved by BBP in its own area, but BBP can then not be called simultaneously by more than one activity and have reliable register contents returned.

The FCT may be generated with all necessary initial information by the following proc call, where * indicates fixed block size:

```
FILE   'filename','BBP' 'SIZE',[*] [max-block-word-count] ;
       'POOL',addr-of-buffer-packet[,look-ahead] ;
       ['ERROR',BBP-call-error-exit,I/O-error-exit,abnormal-condition-exit] ;
       ['SENT',sentinel-value,sentinel-pos,mask-for-block-sentinel-check] ;
       ['LABEL',nbr-of-words,addr] ;
       ['FREEWD',nbr-of-words,addr] ;
       ['REG',8-word-register-save-location]
```

BBP consists of ten basic subroutines (including BJOIN$ and BPOOL$, which may also be used apart from BBP), each of which is entered through the calling sequence:

```
L,U     A0,pktaddr
LMJ     X11,subroutine-name
```

Most of these calling sequence instructions may be generated by proc calls. When this is done, the associated packets are also generated; the packets are put under a different location counter number (30) than the one currently in control. The subroutine names, packets, and descriptions are:

BCLOF$

Close out a file control table (FCT) for an input, an output, or an in/out file, and release the pool of buffers which was held for this file control table.

```
BCLOSE 'FILE' FCT-addr[,'N']
```

To close several file control tables, a string of several fields may be included in this proc call, each naming a different FCT address, with or without option.

When the file is to be freed after being closed, the following proc call generates a test (TZ,S1 FCT-addr+6) and a jump (J $+3) instruction (prevents BCLOF$ from being called if the file is already closed); the two-instruction BCLOF$ calling sequence; the necessary instructions to dynamically @FREE the file; the BCLOF$ packet; and the @FREE image that is used by CSF$. This is everything needed to close and free a tape file:

```
BREL FCT-addr
```

BCLOR$

Close out the current reel, and initialize the following reel, of an input, an output, or an in/out file.

This has the same packet as shown for BCLOF$. The proc call is also the same as that shown for BCLOF$, except that 'REEL' replaces 'FILE':

```
BCLOSE 'REEL' FCT-addr,option
```

BJOIN$

Add additional words to the buffer area that was originally set up by the BPOOL$ subroutine (or through the B$GPUL proc call, which is described under BPOOL$).

WORD	H1	H2
0	0	BPOOL$-packet-or- B$GPUL-call-addr
1	word-length-of-additional-area	addr-of-additional-area

BMARK$

Write a hardware end-of-file mark on an output mode tape file.

```
BMARK FCT-addr
```

Several FCT address fields may be included in this proc call.

BOPEN$

Initialize the file control table for subsequent block buffering operations on an input, output, or in/out mode file.

```
BOPEN 'file-mode-mnemonic' FCT-addr,option
```

To open several files, several FCT address fields may be named in this proc call, with or without option.

BPOOL$

Set up a buffer pool for use by block buffering.

The size of each buffer used by the block buffering subroutines must be three words larger than the maximum size block in the file(s) for which this buffer pool is used.

Instead of using the BPOOL$ subroutine at execution time to organize the buffer pool, the buffer pool may be set up at assembly time by using the following proc call at the starting location of the area which is available for buffers:

B$GPUL number-of-buffers, size-of-each-buffer

BREAD$

Read sequentially the next block from an input or in/out mode file.

BREAD FCT-addr [move-to-area-word-count,move-to-area-addr]

BRREAD$

Read (randomly) the fixed-length block, whose number is specified, from an input or in/out mode FASTRAND-format file.

BRREAD FCT-addr block-number [move-to-area-word-count,move-to-area-addr]

If exclusive use is wanted, use BXREAD instead of BRREAD.

BRWRT$

Writes (randomly) a fixed-length block, into the specified block number position of an output or in/out mode FASTRAND-format file.

BRWRIT FCT-addr block-number [move-from-area-word-count,move-from-area-addr]

BWRIT$

Write sequentially a block into an output or in/out mode file.

BWRIT FCT-addr [move-from-area-word-count,move-from-area-addr]

DYNAMIC DUMPS

Following a program's termination, all information saved by the Dynamic Dump routines is automatically printed out.

Since the primary purpose of these subroutines is to write data into the file DIAG$, it would be impractical for them to be re-entrant. Therefore, not more than one activity of a program should currently reference these subroutines.

There is a word in the data area of the Dynamic Dump routines called XSTAT$ which is initially set nonzero. If it should become desirable for all Dynamic Dump routines to return control immediately, without producing any dumps, XSTAT$ may be cleared to zero. This effectively turns off the Dynamic Dump routines.

SZ XSTAT$ This instruction is generated by the proc call:

X$OFF

To return XSTAT$ to its original nonzero status, the following three instructions (which do not depend upon any specific register contents) may be used:

```
S       A0,XSTAT$
TNZ     XSTAT$
SN,H2   A0,XSTAT$
```

These three instructions (or their equivalent, in some systems) are generated by the proc call: X$ON

The parameter 'format' specifies a single letter, enclosed in quotes, which references either a standard or a user-defined editing format. If omitted, an octal dump is produced.

Format Parameter	Definition	Number of Items Per Line	Number of Print Positions Per Item	Number of Decimal Places
'A'	Alphanumeric	16	6	–
'D'	Double precision floating point	4	24	18
'E'	Floating decimal	8	13	8
'F'	Fixed decimal	8	12	8
'I'	Integer	8	12	–
'O'	Octal	8	12	–
'S'	Instruction	4	20	–

Only the more commonly used dynamic dump routines are described here. Refer to the *UNIVAC 1100 Series Operating System Programmer Reference, UP-4144* (current version), Section 11, for a more detailed description of all the routines.

'format' — Specifies a single letter, enclosed in quotes, which references one of the following standard editing formats: A, E, F, I or O. Standard formats D and S and user-defined formats cannot be specified. If omitted, an octal dump is produced.

The number of calls on X$CW is not limited, but only five separate areas may be dumped.

XDRUM$

Dumps portions of FASTRAND-formatted mass storage by making temporary use of a previously defined buffer initialized by the X$BUFR procedure. Portions of mass storage to be dumped are read into the buffer, then the contents of the buffer is written into the diagnostic file.

```
SLJ XDRUM$
+   word-count,location-addr
+   'format',I/O-pktaddr
```

This linkage may be generated by the procedure call:

X$DRUM I/O-pktaddr,location-addr,word-count,format

I/O-pktaddr — Specifies the address of the I/O packet containing the internal filename.

location-word-addr — Specifies the address of a word which contains the relative starting sector address or a word address of the file to be dumped. (In some cases, this address may be I/O-pktaddr+5, which contains a sector address or a word address.) The manner in which the file was assigned determines whether the address specified is a word address or a sector address.

Use of the X$DRUM procedure requires a main storage buffer into which the mass storage dump can be read. For FASTRAND-formatted files, it is recommended that the buffer be some multiple of 28, the length of a FASTRAND mass storage sector. While a portion of mass storage that is larger than the size of the buffer may be dumped, greater efficiency results by providing a buffer that is sufficiently large to hold all the mass storage to be dumped at one time.

If a main storage buffer is not reserved and initialized for the X$DRUM procedure, no mass storage dump occurs.

The same buffer area can be used for both X$DRUM and X$TAPE procedure calls.

XBACK$ and XMARK$

These routines are used in conjunction to mark the points in program execution between which dynamic dumps are saved and then deleted at the user's discretion. The X$MARK and X$BACK procedures permit a user program under checkout to include dynamic dump procedures which the user may want to print only when a routine does not terminate normally.

FORMAT 1:

```
SLJ XMARK$
```

This instruction may be generated by the procedure call:

X$MARK

FORMAT 2:

```
SLJ  XBACK$
```

This instruction may be generated by the procedure call:

X$BACK

The X$MARK and X$BACK procedures behave much as left and right parentheses surrounding portions of a program which are to be dumped only if termination occurs between them.

X$MARK and X$BACK pairs may be nested to a depth of five. The total number of occurrences of X$MARK and X$BACK is unrestricted.

XBUFR$

Internally defines an area of main storage for use as a buffer by the X$TAPE or X$DRUM procedures.

```
SLJ  XBUFR$
+    word-count,starting-addr
```

This linkage may be generated by the procedure call:

X$BUFR starting-addr,word-count

starting-addr — Specifies the starting main storage address of the buffer.

word-count — Specifies the number of locations in the buffer to be initialized.

XCORE$

Produces a dump of the specified main storage area.

```
     SLJ    XCORE$
N$   FORM   4,14,18
     N$     index-reg,word-count,
            starting-addr
     +      'format',0
```

This linkage may be generated by the procedure call:

X$CORE starting-addr,word-count,'format',index-reg

starting-addr — Specifies the main storage starting location of the dump. If omitted, starting location of zero is assumed.

word-count — Specifies the number of locations to be dumped (37777_8 maximum).

index-reg — Specifies the index register used to modify the address specified by the starting-addr parameter. This parameter, which may be omitted or left zero, can be set to values from 1 to 15 to specify an index register from X1 through A3. The value in the index register is added to the starting-addr value to get the actual dump starting address.

XCREG$

Dumps specified user control registers. (The A,X and R registers and the unassigned registers at addresses 34_8 and 35_8.)

```
SLJ  XCREG$
+    register-count,starting-reg
+    'format',0
```

This linkage may be generated by the procedure call:

X$CREG starting-reg,reg-count,'format'

starting-reg — Specifies the address of the first control register to be dumped.

reg-count — Specifies the number of control registers to be dumped.

XCW$

Produces a changed word dump of specific locations within main storage. On the first X$CW call referencing a given main storage area, a complete dump of that area is produced. On subsequent X$CW calls to the same area, only those words which were changed since the last X$CW procedure call are dumped showing the previous contents and the current contents.

```
SLJ  XCW$
+    word-length,starting-addr
+    'format',0
```

This linkage may be generated by the procedure call:

X$CW starting-addr,word-length,'format'

starting-addr

Specifies the main storage starting location of the dump.

word-length

Specifies the number of locations to be dumped (37777_8 maximum).

XDUMP$

Produces a dump of the program environment, A,X, and R registers, and main storage. Identical to XCORE$ except that a field is provided for specifying A,X, and R registers.

```
     SLJ    XDUMP$
N$   FORM   4,14,18
     N$     index-reg,word-count,
            starting-addr
     +      'format',register-code
```

This linkage may be generated by the procedure call:

X$DUMP starting-addr,word-count,'format','AXR',index-reg

'AXR'

Specifies, enclosed in quotes, one or more letters representing the A,X and R registers. The contents of these registers are printed in octal.

register-code

Register codes for XDUMP$ are:

No registers	0_8
R only	200401_8
A only	200202_8
R and A	400603_8
X only	200104_8
X and R	400505_8
X and A	400306_8
A,X, and R	600707_8

The printout resulting from XDUMP$ is preceded by the heading: **DUMP**. The following additional information is provided following the **DUMP** heading:

element name, location counter, relative program address, and hardware fault indicators.

XFRMT$

Specifies a nonstandard editing format for use by the diagnostic dump procedure calls as an alternative to the standard editing formats or redefines the standard editing formats. New format labels such as 'U', 'V', or 'W' may be specified, or existing standard format labels may be redefined.

```
SLJ  XFRMT$
+    format-specification-word-
     length,'format-label'
'(format-specification)'
```

This linkage may be generated by the procedure call:

X$FRMT format-specification-word-length,'format-label' '(format-specification)'

format-specification-word-length

Specifies the number of words comprising the format specification.

'format-label'

Specifies a single letter enclosed in quotation marks referencing one of the standard editing formats: A,D,E,I,O, or S. This action is used to redefine the standard editing formats. To specify a user-defined editing format, any letter (enclosed in quotes) except A,D,E,I,O, or S may be used.

'(format-specification)'

Specifies a string of alphanumeric characters which represent an encoding of the format to be applied to the information printed. The string of alphanumeric characters may not contain intervening blanks. The first nonblank character of the string must be a left parenthesis (preceded by a quotation mark); the last nonblank character must be a right parenthesis (followed by a quotation mark).

The format of the string of characters that comprises this parameter is specified exactly as in FORTRAN V FORMAT statements. For example, specifying '(10F8.3)' indicates that

the dump information printed on one line consists of 10 words of fixed-point decimal data and that each word is eight characters long with the decimal point at the left of the third least significant character.

Any standard or used-specified editing format may be redefined; the most recent definition prevails.

Multiple line formats are allowed.

Except as indicated below, any format that can be given in a FORTRAN V FORMAT statement can be specified. See *UNIVAC 1100 Series FORTRAN V Programmer Reference, UP-4060* (current version) or *UNIVAC Fundamentals of FORTRAN Programmer Reference, UP-7536* (current version).

The following FORTRAN V editing codes are not supported by PMD: G, J, and R. The editing routines of PMD support an S editing code which assumes that the words being dumped are 1100 Series instructions and are split up into component parts. Twenty spaces in the print line are required for one word. The A editing code assumes that the words being dumped are assumed to contain Fieldata characters.

XMESG$

Permits the user to place any message he desires into the dynamic dump.

```
SLJ  XMESG$
+    word-length-of-msg,'A'
'diagnostic-msg'
```

This linkage may be generated by the procedure call:

```
X$MESG  word-length-of-msg
'diagnostic-msg'
```

word-length-of-msg — Specifies a number equal to the number of computer words in the message (one computer word holds six characters).

'diagnostic-msg' — Any string of alphanumeric characters enclosed in quotes and printed exactly as assembled.

'A' — Generated by the procedure call. It is of no significance to the user, but it must be coded when the instruction form of the format is used.

The X$MESG procedure produces a line on the output listing of up to 120 alphanumeric characters. The printed line immediately follows the procedure reference.

The X$MESG procedure is executed only when the conditional dump switch is on.

XSIZE$

Changes the length of the area of the diagnostic file reserved for dynamic dumps.

```
SLJ  XSIZE$
+    length
```

This linkage may be generated by the procedure call:

```
X$SIZE  length
```

length — Specifies the length (in sectors) of the diagnostic file to be reserved for dynamic dumps.

Using this procedure, a user program can dynamically expand or contract the length of the dynamic dump portion of the diagnostic file. If this is not used, a system standard value is assumed for the length of this portion of the file. If this procedure is used, it should be used before executing dynamic dumps to ensure enough space for those dumps taken.

XTAPE$

Dumps the block of magnetic tape data located just prior to the current tape position by making temporary use of a previously defined buffer initialized by the X$BUFR procedure. The magnetic tape is moved backward one block, the block is read, and the number of words specified in the X$BUFR procedure is dumped.

```
SLJ  XTAPE$
+    word-count,buffer-addr
+    'format',I/O-pktaddr
```

This linkage may be generated by the procedure call:

```
X$TAPE  I/O-pktaddr, 'format'
```

I/O-pktaddr — Specifies the address of the I/O request packet for the device handler. This parameter may be the address of a file control table (FCT) as is

used by block buffering and other routines, since the first six words of an FCT are an I/O packet.

Interblock gaps separate the blocks that are recorded on magnetic tape each time an I/O write of any size word count is done. These interblock gaps serve as block separators. The X$TAPE procedure causes a move backward to the preceding interrecord gap, then a read of everything which follows into the buffer initialized by an X$BUFR procedure until the next interrecord gap is encountered. When the buffer is filled, the remaining words are lost.

The X$TAPE procedure is useful for dumping a block that was just read or written. No dump occurs if the magnetic tape is positioned at the load point (beginning-of-tape marker) or at the interrecord gap following an EOF mark.

No magnetic tape dump occurs if a main storage buffer is not reserved and initialized for the X$TAPE procedure.

The same buffer area can be used for both X$DRUM and X$TAPE procedure calls.

The word count and buffer address are returned by the X$TAPE procedure to the first parameter word.

PROGRAM TRACE ROUTINE (SNOOPY)

SNOOPY is a program trace routine which is designed for use primarily with assembly-language programs. In batch mode, SNOOPY provides a straightforward account of every instruction executed and its effect. In the demand mode, SNOOPY acts as a powerful diagnostic routine affording user control over the trace operation. When used on level 27 and earlier Operating Systems, the tag BANK$ may appear as undefined; this will cause no harm.

Two formats are available for calling SNOOPY:

```
SLJ  SNOOPY
+    mode-bits,termination-addr   mode-word
```

and

```
SLJ  TON$
```

When the first format is employed, tracing begins with the instruction following the mode-word. Tracing continues until the termination address (termination-addr) is reached or until another termination condition is encountered.

When the second format is employed, tracing begins following the SLJ instruction and continues until a termination condition is encountered; quarter-word or third-word mode is determined by the mode set on entry.

When operating in the batch mode, tracing may be terminated by:

(1) Reaching the specified termination address (program execution continues).

(2) Executing an SLJ TOFF$ instruction (program execution continues). If an SLJ TOFF$ instruction is executed outside of the trace routine, it has no effect.

(3) Performing an ER EXIT$. This not only terminates SNOOPY but it also terminates the activity being traced.

(4) Encountering a program contingency of types 1_8, 2_8, 7_8, or 12_8 for which standard system action has been specified. The activity being traced is terminated by EXIT$.

When operating in the demand mode, tracing may be terminated by the following methods in addition to those available for batch mode:

(1) Using the TOFF$ command; program execution continues.

(2) Using the EXIT$ command. This not only terminates SNOOPY but it also terminates the activity being traced.

All commands listed may be abbreviated to the first three characters; all commands except ALTPRT, TON$, RBK and STEP may further be abbreviated to the first character only.

Certain commands (TOFF$, EXIT$, CHANGE) clear SNOOPY's command buffer before reading further commands because of the potentially irreversible nature of the operation to be performed. If this is not desired, an asterisk may be affixed to the command, as TOFF$*, EXIT$*, CHANGE*. For example, to terminate a trace and continue an execution without typing in two lines, the sequence 'TOFF$* GO" may be used.

Command	Description
ABSAD	Convert relative program addresses to absolute addresses. The three parameters are: eltname,loc-counter,location. The ABSAD command may be used to determine the value assigned by the collector to a particular externally defined symbol. Only referenced symbols defined by non-RLIB$ routines may be looked up unless the collection was done using the "TYPE EXTDIAG" statement, in which case all symbols may be looked up. The command format is "ABSAD <symbol> *" where the trailing asterisk indicates that an external symbol is meant. For example, "ABSAD SNOOPY*" will print the address of SNOOPY's entry point "SNOOPY".
ALTPRT	Send all trace printout to an alternate print file, while command solicitation, command responses (as by DUMP, for example) and all print requests by the program are sent to the terminal as usual. One parameter may be given, which is the name of the file to be used as an alternate print file. When the ALTPRT command is given and an alternate file is in use, the current alternate file will be @BRKPTed. This action may be suppressed by employing an asterisk (*) as the trailing delimiter for the parameter; e.g., either "ALTPRT <filename> *" or "ALTPRT *". To obtain printout both at a terminal and in an alternate file, the ALTPRT filename should have a trailing exclamation point (!); e.g., "ALTPRT <filename> !". If an alternate file is already active, the command forms "ALTPRT !" and "ALTPRT ?" may be used to set and clear echo mode, while leaving the same alternate file in use. Echo mode is always cleared when the trailing exclamation point is not used; therefore, to start a new alternate file without breakpointing the old file and with echo mode set, the commands "ALTPRT * ALTPRT <filename>!" must be used.
BREAK	Automatically return to command mode when control reaches a specified point in the program. Only one breakpoint may exist at a time. The three parameters are: eltname,loc-counter,rel-addr
CHANGE	Allows the user to change the contents of control registers or main storage. The single parameter gives the location to be changed. If the parameter is a register name, a number, or a number preceded by an H or Q, the new value is to be entered as a single octal number. If the parameter is a number preceded by the letter I, the new value to be entered consists of six numbers, each separated by a space, and representing the f, j, a, x, hi, and u fields of an instruction word. The CHANGE command allows the use of mnemonics and external symbols for I-format (instruction format) changes, as well as octal values. The first item given to the "NEW VAL-:" type-out may be an op-code mnemonic instead of an octal number for the F-field. Abbreviated forms such as L, LN, ANM are not permitted, however; LX, LA, LR, LNA, ANMA and so on, must be used. For some instructions, the op-code mnemonic specifies values for the J-field and perhaps the A-field as well. In such cases, the next value given to the change command will be an A-field or X-field. Mnemonics may also be used for J-designator values and for standard X-, A-, and R-register names. If a register name is used in the A-field of an instruction, its value will be adjusted appropriately. Truncation errors are not detected. Fields of an instruction are always expected to be entered in the order F, J, A, X, HI, and U. Note that the H and I fields are combined. An external symbol may be used for any field except the F-field, subject to the same restrictions as for the ABSAD command.

Command	Description
DUMP	Display the program status. Each parameter must be separated by a comma. If no parameter or an empty parameter (that is, two consecutive commas) is given, all registers and the carry and overflow designators are dumped. The parameters are: A,X, or R – Dumps the indicated group of registers. To dump the contents of a single register, use the register mnemonic or the octal address. T – Dumps the carry and overflow designators. The letter I preceding a number produces an instruction-format dump. The letter H preceding a number produces a Fieldata-character dump. The letter Q preceding a number produces an ASCII-character dump. B – Display names of active banks (in order main-I, main-D, utility-I, utility-D, with commas indicating place for unbased PSR portions). E – Last contingency, if any. L – Display current storage limits. S – Display names of active segments. Combinations such as HA0, IA13, QR7 are acceptable and are interpreted correctly. Any dump specification which references an address may be given a trailing + sign followed by an octal number N. A dump is then taken of the N consecutive storage locations following the original address, in the same format as the first dump, resulting in a dump of N+1 locations.
EXIT$	Terminates the traced activity by means of an EXIT$ request. Trace mode is terminated and the last instruction is printed.
GO	Return to trace mode from command mode.
JHT	A jump history table is maintained by SNOOPY; this table contains the addresses of the last 8 jump instructions which caused a transfer of control. The JHT command will cause this table to be printed, starting with the most recent jump-from address. The table is cleared on entry to SNOOPY, so fewer than 8 addresses may be printed early in a trace.
JUMP	Transfers control to a specified absolute address. The current absolute and relative P register values are displayed. If the new value is within the program storage limits, that value is set into the P register. The new value is printed in relative form and the next command is executed. The jump-to address specified for the JUMP command may be an external symbol as well as an octal value, subject to the restrictions on externally defined symbols noted for the ABSAD command.
LINE	Adjust the length of the line printed by SNOOPY. If "LINE /" is entered, the line length set is the default value of 110_8. Otherwise, the parameter given to LINE must be an octal number denoting the line length for the device in use.
number	Has the same effect as a SKIP n GO sequence (where n is the number). See SKIP command. The number is in octal.
PRINT	Allows modification of the amount of printing. The PRINT command recognizes only one parameter at any one time. If an invalid parameter is specified, the F parameter is assumed. The parameters are: C – Produce a printout omitting extraneous spaces used for formatting. E – Produce an expanded printout (formatting spaces are included). The E mode is effective until a PRINT C is encountered. F – Produce a full printout consisting of each instruction, its location, and the contents of main storage and registers (in before/after form if the value changed). For certain Executive Requests, the contents of the associated packet is also dumped. This is the default mode.

Command	Description
	N – Suppresses printout. This provides a means of skipping long sections of irrelevant code. P – Produce printout of the instructions but not referenced main storage or Executive Request packets. If SNOOPY is in the N mode, the P mode is set automatically upon the occurrence of an RBK contingency or encountering a BREAK specified break condition.
RBK	Allows the user to simulate an RBK contingency for the executing program; the actual RBK contingency is intercepted by SNOOPY and directs a return to command mode. This command provides the means for tracing a contingency routine. If the user program does not expect the contingency, an appropriate message is displayed.
RELAD	Convert absolute program addresses to relative addresses. The only parameter is: location. Ambiguities are resolved in favor of elements residing in currently loaded segments in currently active banks.
RLIB	"RLIB L" will print the system type, system level identification, and site as well as the RLIB$ level used at collection time. "RLIB E, <element-list>" where <element-list> is a list of element names separated by commas specifies that the elements named are to be treated as if they were RLIB$ elements for the purposes of RLIB$ trace suppression. The list specified completely replaces any preceding list. An empty list is specified by following the "E" with punctuation other than a comma or question mark. "RLIB E?" prints the current list. At present, the list may contain at most 16 names.
SKIP n	Return to command mode after executing n number of instruction cycles. If n is omitted, any previously existing skip count is deleted and no skip interrupt occurs. Otherwise, an octal number is used to set the interrupt point. If the count is exceeded during an indirect addressing or execute remote cascade, the command mode is re-entered when the instruction is completed.
STEP	Execute one instruction in trace mode and return to command mode.
TOFF$	Leave the trace mode and continue execution as if an SLJ TOFF$ command had been executed. Trace mode is terminated and the last instruction is printed.
TON$	Restart a trace that was to be terminated and execute one instruction. To compute the number of instruction cycles performed, use the TOFF$ command followed by the TON$ command. The TON$ command is not affected if the activity is about to terminate by means of an EXIT$ request; if it is desired to continue tracing from that point, a JUMP command must first establish a point from which execution will continue.
TRAP	A new command is available which will cause SNOOPY to enter command mode from trace mode whenever one of a set of locations is referenced or altered, except for ER operations. Entering the command "TRAP <loc–1>, <loc–2> . . ." will place up to sixteen locations in the trap table. Commas must be used as separators. The locations may be octal numbers, register mnemonics, or external symbols. The use of external symbols is subject to the restrictions noted for the ABSAD command. Each list specified completely replaces the preceding list. The command "TRAP?" will print out the current list.

EDITING PACKAGE FOR IMAGE COMPOSITION (EDIT$)

EDIT$ is a set of simplified re-entrant subroutines which may be used for composing strings of Fieldata characters in an area specified by the user. It is particularly useful in preparing images for ER CSF$, ER PRINT$, ER PUNCH$, ER PRTCN$, and PCHCN$.

EDIT$ works from the following packet, where words 6 through 9 are not needed unless there are floating-point (FP) numbers to be edited.

WORD	S1	S2	S3	S4	S5	S6
0	[test-and-set]	EMSG$-stop	image-length	image-addr		
1	*char-index*	*word-index*	*EMSG$-char*	*EMSG$-word(index)*		
2	FPS	FPR	0	*return-addr-for-char-store*		
3	*user's-return-addr*			*save-of-original-X1-modifier*		
4	*save-of-original-X2-contents-or-save-of-character-pointer*					
5	*save-of-original-X3-contents-or-save-of-word-pointer*					
6	[DPC]	[SPC]	*digits-before*	*digits-after*	*negative-sign*	*not-normalized*
7	*final-column-position*			*characteristic's-power-of-ten*		
8	*save-area-for-intermediate-floating-point-results*					
9						

Most of these packet fields, shown in italics, do not require attention, and are used simply for internal control purposes by EDIT$.

- When not using EDIT$'s floating-point subroutines, the user needs to supply information for only three fields, which are all located in word 0 of the packet:

  EMSG$-stop

  A special stop character ('&' is frequently used) which, when one or more are imbedded in an EMSG$ input image, causes the EMSG$ routine to stop transferring characters from the input image each time it encounters one of these characters. This can serve not only as the final EMSG$ input image stop, but can also cause intermediate stops to allow the user to insert new information (such as status codes or names, using various other EDIT$ routines) at any predetermined points within the EMSG$ image.

  image-length

  The size of the output image buffer (normally 22 words when composing lines of print for a 132-column line printer).

  image-addr

  The address of the output image buffer. When the image is fully formed, this buffer can be directly referenced on a call to CSF$, PRINT$, etc.

- Four additional fields are used only with floating point:

  FPS

  The scale, or number of digits to be placed before the decimal point, for scientific format floating-point editing. This is usually set to 1.

  FPR

  Set nonzero (which is the usual case) to specify floating-point rounding, which means that 5 is added to the eighth significant digit for single-precision numbers or to the eighteenth significant digit for double-precision numbers.

  DPC

  If nonzero, this is the character to use when editing double-precision floating-point numbers, to separate the mantissa and the characteristic. (This is for compatibility with existing formats, such as the 'E' used with FORTRAN.)

  SPC

  This is the same as DPC, except that it is for single-precision floating-point numbers.

- The following proc call generates a six-word EDIT$ packet, where '&' is generated for the EMSG$-stop-character if the 'MSG' parameter is omitted:

  E$PKT image-length,image-addr ['MSG','EMSG$-stop-character']

- The following proc call generates a 10-word EDIT$ packet, where these assumed values are inserted in omitted parameters: 'MSG'='&', 'FPS'=1, 'FPR'=1, 'DPC'=0, 'SPC'=0.

E$PKTF image-length,image-addr ['MSG','EMSG$-stop-character'] ;

['FPS',FPS-number] ['FPR',FPR-number] ['DPC','DPC-character'] ;

['SPC','SPC-character']

On all of the proc calls shown below for EDIT$ subroutines, when any of the parameters are omitted, the corresponding calling sequence's load A0 or load A1 instruction is not generated.

- EDIT$ has three subroutine calls for initiating or terminating editing mode:

EDIT$

Initiate editing mode. The contents of registers X1, X2, and X3 are saved. The image is space-filled, and the column pointer is set to the start of the image. (Note that EDIT$ also uses, but does not save or restore, registers X11, A0–A3, and R1.)

```
L,U     A0,pktaddr
LMJ     X11, EDIT$
```

These two instructions are generated by the proc call:

E$DIT pktaddr

EDITX$

Terminate editing mode. The column pointer is saved in the packet. Registers X1, X2, and X3 are restored to their original contents. The address of the packet that was active is returned in A0.

```
LMJ     X11,EDITX$
```

This instruction is generated by the proc call:

E$DITX

EDITR$

Re-establish the editing mode to its status at the time of the previous call to EDITX$. The column pointer saved by EDITX$ is restored.

```
L,U     A0,pktaddr
LMJ     X11,EDITR$
```

These two instructions are generated by the proc call:

E$DITR pktaddr

- EDIT$ has 18 nonfloating-point subroutines for such purposes as converting numbers into Fieldata, inserting strings of characters into the image, and manipulating the column pointer. Note that in all cases where one or more characters are inserted into the image, they are inserted beginning at the current column pointer location (which is initially set to column 0 by the initiate editing mode subroutine), and the column pointer is always advanced to the column following the last inserted character. These subroutines are listed alphabetically:

ECHAR$

Insert the character in S6 of A0 into the image.

```
L,U     A0,'character'
LMJ     X11,ECHAR$
```

These two instructions are generated by the proc call:

E$CHAR 'character'

or

```
L       A0,addr-of-character
LMJ     X11,ECHAR$
```

These two instructions are generated by the following proc call, where addr-of-character may contain any of the x, h, i, u, or j instruction word fields in the standard *u,*x,j form (see Section 6):

E$CHAR addr-of-character

ECOL$

Advance or back up the column pointer to the column number in A0. (Note that a previously inserted character or string can be backed up to, and then overwritten if desired.)

```
L,U     A0,column-number
LMJ     X11,ECOL$
```

These two instructions are generated by the proc call:

E$COL column-number

ECOLN$

Obtain the current column pointer number in A0.

```
LMJ     X11,ECOLN$
```

This instruction is generated by the proc call:

E$COLN

ECOPY$

Insert into the image the number of characters specified in A1, taken from the location whose starting address is given in A0. All characters are transferred, including spaces.

```
L,U   A1,character-count
L,U   A0,addr-of-characters
LMJ   X11,ECOPY$
```

These three instructions are generated by the following proc call:

```
E$COPY  character-count,addr-of-characters
```

EDAY1$

Convert the date portion of a TDATE$ format word (see ER TDATE$ – Section 2) in A0 to an eight-character Fieldata string of the informal-U.S.-style date representation where, for example, '02/04/70' would represent the 4th of February, 1970. Insert this string into the image.

```
L     A0,addr-of-TDATE$-word
LMJ   X11,EDAY1$
```

These two instructions are generated by the proc call:

```
E$DAY1  addr-of-TDATE$-word
```

or, for today's date:

```
ER    TDATE$
LMJ   X11,EDAY1$
```

These two instructions are generated by the proc call:

```
E$DAT1
```

EDAY2$

This is the same as EDAY1$, except that the date cited by the example would be converted to the unambigious nine-character Fieldata string, '04 FEB 70'. The corresponding proc calls for EDAY2$ are E$DAY2 and E$DAT2.

EDAY3$

This is the same as EDAY1$, except that the date is converted to a variable length string, ranging from 11 to 18 Fieldata characters, defined briefly by the following three examples: 'FEBRUARY 4, 1970', 'MAY 1, 1970', 'SEPTEMBER 18, 1970'. The corresponding proc calls for EDAY3$ are E$DAY3 and E$DAT3.

EDECF$

Convert the number in A0 to Fieldata decimal digits, and set the result (right-justified and space-filled) into a fixed-length field of the number of characters specified in A1. If the number of Fieldata digits (including a leading '–', if the number in A0 is negative) exceeds the field size specified, they overflow the field. Insert this field into the image.

```
L,U   A1,character-count
L     A0,addr-of-number
LMJ   X11,EDECF$
```

These three instructions are generated by the proc call:

```
E$DECF  character-count,addr-of-number
```

EDECV$

This is the same as EDECF$, except that the field size is variable, and only as large as is necessary to hold the converted Fieldata decimal number (including a leading '–', if the number in A0 is negative).

```
L     A0,addr-of-number
LMJ   X11,EDECV$
```

These two instructions are generated by the proc call:

```
E$DECV  addr-of-number
```

EFD1$

Insert the contents of A0 into the image, excluding any sixth word in A0 whose value is zero (Fieldata at sign '@') or five (Fieldata space, ' '). This can be used to insert a Fieldata name of one word or less, where a partial word filler of zero or Fieldata spaces is not wanted in the image.

```
L     A0,'Fieldata-name'
LMJ   X11,EFD1$
```

These two instructions are generated by the proc call:

```
E$FD1  'Fieldata-name'
```

EFD2$

This is the same as EFD1$, except that it is for a Fieldata name of two words or less, contained in A0 and A1.

```
DL    A0,'Fieldata-name'
LMJ   X11,EFD2$
```

These two instructions are generated by the proc call:

E$FD2 'Fieldata-name'

EMSG$

Insert the characters starting at the address given in A0 into the image. This process stops when the character in S2 of word 0 of the EDIT$ packet is encountered in the EMSG$ input string. The pointer for the EMSG$ input string is saved in the packet, for possible further use on an EMSGR$ (EMSG$ reentry) call.

```
L,U   A0,input-string-addr
LMJ   X11,EMSG$
```

These two instructions are generated by the proc call:

E$MSG input-string-addr

EMSGR$

Re-enter the EMSG$ subroutine, and begin copying from the EMSG$ input stream following the point of previous interruption. With the EMSG$ and EMSGR$ subroutines, it is possible to copy a string into the image, occasionally interrupting this action to perform other EDIT$ functions at certain selected points in the string.

```
LMJ   X11,EMSGR$
```

This instruction is generated by the proc call:

E$MSGR

EOCTF$

Convert the number in A0 to Fieldata octal digits, and set the result (right-justified and zero-filled) into a fixed-length field of the number of characters specified in A1. If the number of Fieldata octal digits (including a leading '0', if the magnitude of the number is greater than 7) exceeds the field size specified, truncate one or more high order digits. Insert this field into the image. Note that EOCTF$, unlike EDECF$, does not assume that the number in A0 is necessarily a representation of a single positive or negative quantity, and therefore EOCTF$ does not take special action to complement the number and add a leading '–' sign if the high order sign bit is set. If, for example, A0 had all bits set to 1, except bit 0, EOCT$ would convert this to the 13-character string, '0777777777776', while EDECF$ would convert it to a 2-character string of '–1'.

```
L,U   A1,character-count
L     A0,addr-of-number
LMJ   X11,EOCTF$
```

These three instructions are generated by the proc call:

E$OCTF character-count,addr-of-number

EOCTV$

This is the same as EOCTF$, except the field size is variable, and only as large as is necessary to hold the converted Fieldata octal number (including a leading '0', if the magnitude of the number is greater than 7).

```
L     A0,addr-of-number
LMJ   X11,EOCTV$
```

These two instructions are generated by the proc call:

E$OCTV addr-of-number

EPACK$

This is the same as ECOPY$, except that sixth words whose value is zero (Fieldata at sign, '@'), although included in the input character count, are not inserted into the image.

```
L,U   A1,character-count
L,U   A0,addr-of-characters
LMJ   X11,EPACK$
```

These three instructions are generated by the following proc call:

E$PACK character-count,addr-of-characters

ESKIP$

Advance the column pointer by the column count given in A0. To back up the counter, the column count should be loaded negative into A0.

```
L,U   A0,column-count
LMJ   X11,ESKIP$
```

These two instructions are generated by the following proc call:

E$SKIP column-count

ETIME$

Convert the time portion of a TDATE$ format word (see ER TDATE$ – Section 2) to an eight-character Fieldata string of the form 'hh:mm:ss', where hh, mm, and ss are the two digits of the hours, minutes, and seconds, respectively. Insert this string into the image.

```
L       A0,addr-of-TDATE$-word
LMJ     X11,ETIME$
```

These two instructions are generated by the proc call:

E$TIME addr-of-TDATE$-word

or for the present time:

```
ER      TDATE$
LMJ     X11,ETIME$
```

These two instructions are generated by the proc call:

E$TD

- EDIT$ has six subroutines for floating-point editing. They are all very similar. The three double-precision surboutines, EFLF2$, EFLG2$, and EFLS2$, are not described separately below, since they perform the same operations on a double-precision number (in A1 and A2), as the single-precision subroutines, EFLF1$, EFLG1$, and EFLS1$, perform on a single-precision number (in A1). The double-precision proc calls, E$FLF2, E$FLG2, and E$FLS2, generate a double load of A1, rather than a single-word load instruction.

  On entering any of the floating-point subroutines, A0 must contain a number of the form X*/6+Y, where X is the desired field size and Y is the desired number of significant digits.

  The minimum necessary field size to avoid overflow to the right is Y+4, plus 1 for each of the following three cases: a separator character is used (see EDIT$ packet fields DPC and SPC); the number is negative (requiring a leading '–' character); the number is double precision (which produces a three-digit characteristic) instead of single precision (which produces a two-digit characteristic). When the field size is larger than is necessary, the edited floating-point result is put in right-justified and space-filled.

  EFLF1$

  Convert the floating-point number in A1 to a Fieldata string in fixed-point format, which consists of a leading '–' (if the number is negative), a '.' to represent the decimal point, the number of digits specified by Y, a separator character (if given in packet field SPC), a '+' or '–' to indicate that the characteristic is a positive or negative exponent, and a two-digit characteristic. Set this string into a field of X characters, and insert the field into the image.

  ```
  L,U     A0,X*/6+Y
  L       A1,addr-of-number
  LMJ     X11,EFLF1$
  ```

  These three instructions are generated by the proc call:

  E$FLF1 X*/6+Y,addr-of-number

  EFLG1$

  This is the most generalized floating-point subroutine. It is the same as EFLF1$, except that an attempt is made to shift the decimal point among the significant digits in such a way that the characteristic will go to zero, in which case the characteristic is set to spaces. If this attempt fails, the decimal point is placed following the number of significant digits specified in EDIT$ packet field FPS.

  ```
  L,U     A0,X*/6+Y
  L       A1,addr-of-number
  LMJ     X11,EFLG1$
  ```

  These three instructions are generated by the proc call:

  E$FLG1 X*/6+Y,addr-of-number

  EFLS1$

  This is the scientific format floating-point subroutine. It is the same as EFLF1$, except that the decimal point is always placed following the number of significant digits specified in EDIT$ packet field FPS.

  ```
  L,U     A0,X*/6+Y
  L       A1,addr-of-number
  LMJ     X11,EFLS1$
  ```

  These three instructions are generated by the proc call:

  E$FLS1 X*/6+Y,addr-of-number

ASCII IMAGE COMPOSITION EDITING PACKAGE (AEDIT$)

AEDIT$ is a set of re-entrant subroutines used for composing strings of ASCII characters in a user-specified area. The AEDIT$ package is very similar to the EDIT$ package used for Fieldata images. AEDIT$ is useful in preparing images for:

(1) Printed ASCII output (ER APRINT$)

(2) Punched ASCII output (ER APUNCH$)

(3) Other Executive Requests which require ASCII images

AEDIT$ works from the following packet. Words 7–10 are used for editing floating-point numbers only.

WORD	S1	S2	S3	S4	S5	S6
0	[Test and Set]	*qwm*	image length	image address		
1	*character index*	*word index*	*AEMSG$ char.*	*AEMSG$ word (index)*		
2	fps	fpr	unused	*return address for char. store*		
3	*user's return address*			*save of original X1 modifier*		
4	*save of original X2 contents, or save of character pointer*					
5	*save of original X3 contents, or save of word pointer*					
6	AEMSG$ Stop char. (Q1)	[dpc] (Q2)		[spc] (Q3)	unused	
7	unused		*digits before*	*digits after*	*negative sign*	*not normalized*
8	*final column position*			*characteristic's power of ten*		
9	*save area for intermediate floating-point results*					
10						

■ GENERATING THE AEDIT$ PACKET

The following PROC call generates a seven-word AEDIT$ packet (for non-floating-point routines), where msg = '&' if field 2 is omitted.

```
A$EPKT  image-length, image-address ['MSG','AEMSG$-stop']
```

The following PROC call generates an eleven-word AEDIT$ packet (for editing floating-point numbers), where the values msg = '&', tps = 1, fpr = 1, dpc = 0, spc = 0 are inserted if the corresponding field is omitted.

```
A$EPKTF  image-length, image-address ['MSG', 'AEMSG$-stop']
    ['FPS', fps-number]  ['FPR', fpr-number]
    ['DPC', dpc-char]  ['SPC', spc-char]
```

■ ASCII EDITING ROUTINE DESCRIPTIONS

The AEDIT$ routines use the same calling sequences as the corresponding EDIT$ routines, except for the routine names. To get the AEDIT$ routine name, simply add 'A' before the name of the corresponding EDIT$ routine (e.g., ECHAR$ becomes AECHAR$). To get the AEDIT$ PROC name, replace the leading 'E$' of the corresponding EDIT$ PROC name with 'A$E' (e.g., E$SKIP becomes A$ESKIP).

PROC	Routine	Description
A$EDIT	AEDIT$	Initial entry into ASCII edit mode. The image is space-filled; the column pointer is set to column 0; quarter-word mode is set; X1–X3 are saved. The AEDIT$ package uses, but does not save or restore X11, A0–A3, R1.
A$EDITX	AEDITX$	Terminate ASCII edit mode. Restore X1–X3; save column pointer in packet.
A$EDITR	AEDITR$	Re-entry into ASCII edit mode. Column pointer is restored.
A$ECHAR	AECHAR$	Edit an ASCII character. Insert the ASCII character from Q4 of A0 into the image.
A$ECOL	AECOL$	Position the pointer to a fixed column.
A$ECOLN	AECOLN$	Obtain the current column number in A0.
A$ECOPY	AECOPY$	Copy a string into the image.
A$EDAY1 A$EDAT1	AEDAY1$	Edit the date portion of A0 into the format: mm/dd/yy (Use A$EDAT1 for the current date)
A$EDAY2 A$EDAT2	AEDAY2$	Edit the date portion of A0 into the format: dd mmm yy
A$EDAY3 A$EDAT2	AEDAY3$	Edit the date portion of A0 into the format: month dd, year
A$EDECF	AEDECF$	Convert to ASCII decimal (fixed length field).
A$EDECV	AEDECV$	Convert to ASCII decimal (variable length field).
A$EFD1	AEFD1$	Insert ASCII (one word). Insert the contents of A0 (four ASCII characters) into the image, excluding any quarter-word whose value is 040 (ASCII space) or 000.
A$EFD2	AEFD2$	Insert ASCII (two words). This is the same as AEFD1$, except that it inserts the contents of A0 and A1 (eight ASCII characters) into the image.
A$EMSG	AEMSG$	Message editor (initial entry). Insert ASCII characters starting at the address in A0 into the image. This process stops when the AEMSG$ stop character (Q1 of word 6 of the packet) is encountered in the string.
A$EMSGR	AEMSGR$	Message editor (re-entry).
A$EOCTF	AEOCTF$	Convert to ASCII octal (fixed length field).
A$EOCTV	AEOCTV$	Convert to ASCII octal (variable length field).
A$EPACK	AEPACK$	Copy and pack a string into the image. This is the same as AECOPY$, except that the quarter-word whose value is 000, although included in the character count, is not inserted into the image.
A$ESKIP	AESKIP$	Skip an area in the image (advance column pointer).
A$ETIME A$ETD	AETIME$	Edit the time portion of A0 into the format: hh:mm:ss (Use A$ETD for the current time)

PROC	Routine	Description
A$EFLF1	AEFLF1$	ASCII single-precision fixed-point format. A1 is edited to fixed-point format with y digits following the decimal point, right-justified, in a field of size x, where the calling sequence is: L,U A0,x*/6+y L A1, addr-of-number LMJ X11, AEFLF1$
A$EFLG1	AEFLG1$	ASCII single-precision generalized format.
A$EFLS1	AEFLS1$	ASCII single-precision scientific format.
A$EFLF2	AEFLF2$	ASCII double-precision fixed-point format. This is the same as AEFLF1$, except that the contents of A1 and A2 are edited.
A$EFLG2	AEFLG2$	ASCII double-precision generalized format.
A$EFLG2	AEFLS2$	ASCII double-precision scientific format.

4. GENERAL ERROR AND STATUS INFORMATION

ERR MODE (EMODE) AND I/O STATUS CODES

This set of error codes is categorized as being under contingency type 12_8.

Most of these codes relate to errors users make when setting up Executive Requests (ERs). The most common user errors are improperly set up, improperly referenced, and inadvertently overwritten packets.

The following list is the full set of defined ERR mode codes, with two exceptions:

- Type 1 (I/O) codes 0_8 through 17_8 and 40_8 are included for the sake of completeness, even though they represent status conditions that are not necessarily errors, and do not directly force a run into ERR mode.
- Types 6 and 7 (communications) codes are not included because they are lengthy and not used by most programmers.

Type	Code Octal	Description
I/O (1)	0	The request has been completed normally. If data transfer is involved, the count is given in H2 of word 4. ADH only – request completed and an interrupt occurred. Normal completion is to be determined by the program.
	1	End-of-file block detected on magnetic tape. (a) Answer of E was given to an I/O error message. (b) End-of-file block was detected on magnetic tape. (c) Block read drum function was truncated by encountering an end-of-block word. (d) Block search read function was truncated by encountering an end-of-block word before the specified number of words were transferred. ADH only – the specified time interval has expired without an interrupt occurring. The operator response to the timeout message is in the A1 register in Fieldata (B, D, E or G) if the timeout field (S3 of word 4) is zero (an A response is not recorded). Otherwise the A1 register is set to zero and no timeout message is displayed.
	2	End-of-tape mark encountered on magnetic tape on a read backward from load point or on a write. No transfer takes place for the read backward. The write is done in the normal manner. Subsequent writes are performed in the same fashion and, barring other problems, result in returning the same status code.
	3	No find was made on a mass storage device search. The search was terminated by an end-of-block, end-of-track, end-of-position, or expiration of sufficient time to pass over the entire area of concern depending upon the physical device and type of search.
	4	A nonintegral block was read from magnetic tape. The number of data characters accepted from the last word is indicated by S3 of word 4 of the packet.
	5	An attempt was made to initiate a mass storage search or read from an area which is wholly or partially unassigned. If the starting address is legal the read is truncated as reflected by the word count in the substatus field. An absolute read request was issued specifying an illegal mass storage address.
	10	The area of the FASTRAND mass storage file being unlocked by this write or unlock request timed out in the locking list. Other requests by other activities for the area may have been honored in the interim. If the function is write, the transfer is not performed.

Type	Code Octal	Description
I/O (1)	11	A nonrecoverable error has occurred. The suppress recovery mode is set for magnetic tape or an answer of G was given to an error message. If the suppress recovery mode is set, the EI status code is stored in A1 of the interrupt activity control register set. All suppress recovery operations come back with this status.
	12	A read, or write error on magnetic tape has resulted in loss of position on the unit. This code is returned for all outstanding requests at the time the answer of B was entered in response to the I/O error message. Any subsequent request is honored but no further program checkpoints are valid. For mass storage devices this status indicates a bad spotted granule was encountered in the file or an answer of B was given to a mass storage I/O error message.
	13	The peripheral unit was declared down by an unsolicited operator keyin.
	20	Some form of write or a function causing area release was attempted on a file assigned in the read-only mode, or a form of read was attempted on a file in the write-only mode.
	21	An attempt was made to reference a filename for which no assignment has been made. ADH only – EF buffer control word length greater or less than that allowed for device being accessed, or the file specified in the packet is not assigned to the program. For MSA devices when one function operation has been found and a second function operation is indicated.
	22	An attempt was made to write beyond the maximum assigned space (via @ASG) for mass storage file or to expand a word-addressable format file when no space is available. ADH only – equipment being referenced is not allowed for ADH interface.
	23	The packet address specified in the A0 register is not within the program limits or defines a packet split between the instruction and data banks of the program, or defines a packet in a write-protected bank.
	24	The function code is not defined for the assigned equipment type. This code also covers noncompatible fields on a set mode request. An absolute read or write was attempted by user program. ADH only – illegal value in command string or, if device being referenced is attached to an MSA, the M field in the MSA command string was found set.
	25	The I/O access word refers to a buffer which is wholly or partially outside of the program area or split between the instruction and data bank of the program. For GW$, SCR$, and SCRB$ functions, this error code is given if the number of access words is 0 or more than 50 or if the total word count is more than 65535. ADH only – if device being referenced is attached to an MSA, a monitor operation was encountered in the string. Otherwise an access word within the packet was found completely or partially outside of program limits.
	26	Illegal starting address given for an interrupt activity. This also covers the case of the last mode being with monitor but no monitor interrupt activity specified.
	27	An I/O request was made with the status word of the request packet set negative indicating a possible program loop.
	31	A magnetic tape operation was issued with user recovery specified and an interrupt activity was not specified (that is, entrance was not made via IOI$, IOXI$ or IOWI$).
	33	A FASTRAND-formatted or word-addressable I/O request may cause the PCT to expand past its maximum. The I/O request is not initiated.

Type	Code Octal	Description
I/O (1)	34	An absolute read or write was attempted on an illegal unit or subsystem.
	35	Errors on read and lock, and unlock requests: (a) A second read and lock (RDL$) request by an activity for a particular area. (b) An unlock (UNL$) request for an area that the activity had not previously locked.
	36	Errors on WAIT$ requests: (a) No I/O is outstanding for this packet. (b) WAIT$ request was not immediately preceded by a Test Positive instruction or the Test Positive instruction had a nonzero h or i field. (c) A WAIT$ or WANY$ request was made without a previous outstanding I/O request for the program. (The executive tests for the case of I/O completion between the time the ER is initiated and the time it is processed without error notification.)
	40	The request is either in the process of being executed or is listed on the request queue for the particular channel.
SYMB (02)	2	Second abnormal return from READ$ or AREAD$.
	3	I/O error (READ$, AREAD$, READA$ or AREADA$).
	4	Image length error.
	5	@ADD error, run stream.
	6	READ$ or AREAD$ access word failure.
	7	Improper SDF control image, READ$, AREAD$, READA$ or AREADA$ request.
	10	Attempt to @ADD an element from tape.
	11	Nested level exceeds maximum (@ADD).
	12	@ADD file not assigned or catalogued.
	13	@ADD element not found in file.
	14	Nested @ADD loop.
	15	@ADD file equipment type error.
	16	Cannot assign @ADD file because PCT is at maximum size.
	20	Alternate file not assigned for demand or real-time run.
	21	Cannot assign punch file or alternate print file.
	22	Type of call does not match type of file.
	23	Alternate packet out of limits.
	24	Read alternate file not assigned.
	25	Error on first read from read alternate.
	26	Alternate file not FASTRAND or tape.
	27	Maximum number of active alternate files exceeded.
	30	Maximum number of breakpoints for print or punch exceeded.
	40	Buffer out of limits.
	41	MAX pages.
	42	MAX cards.

Type	Code Octal	Description
SYMB (02)	43	Illegal syntax in control image.
	44	Maximum length exceeded on control image.
	45	I/O error on creating output file for breakpointed or alternate print/punch file.
	46	Read alternate I/O error.
ER (04)	1	ER index out of range or ER for executive only.
	2	Bad packet limits on ER.
	3	ER index within range, but not in use.
	4	Error encountered on AWAIT$ request.
	5	Bad activity number (id) specified on FORK$ request. Either out of range or already in use.
	6	Account number does not permit requested real-time priority (RT$, FORK$).
	7	Invalid queue pointer or TS cell location encountered in Test and Set Queuing.
	10	FACIL$/FACIT$/FITEM$ packet failed access word check.
	11	FACIL$/FACIT$/FITEM$ I/O error or PCT name section in error.
	20	Bad BBEOF$ packet or invalid file control table address.
	21	File not catalogued or file not mass storage FASTRAND format.
	31	Illegal creation of real-time activity via FORK$ request.
	32	A specified activity name cannot be found in the activity name table on a NAME$, ACT$, INT$, or DACT$ request.
	33	There is already an II activity specified.
	34	Attempt to set quarter-word mode via ER PSR$ when quarter-word mode not allowed on system.
	37	Filename not assigned for tape swap.
	40	Syntax error on CSF$ image.
	41	CSF$ image length is greater than 40 words.
	42	Illegal command for CSF$.
	43	Image is outside user's program limits.
	44	Log entries for this run exceed MAX allowed by CSF$.
	45	ER ABSAD$ not allowed under the run's account number.
	47	Invalid input parameter to LOAD$ (A1,H1 or A2,H1 nonzero).
	50	I/O error encountered when loading segment.
	51	Request to load an undefined segment.
	52	Invalid information in segment load table.
	53	Input parameter contains common BDI, out-of-range BDI, or an address less than the lowest address of the specified bank on ER MCORE$.
	54	Input parameter contains common BDI, out-of-range BDI, or an address less than the lowest address of the specified bank on ER LCORE$.
	55	MCORE$ request for core not available.

Type	Code Octal	Description
ER (04)	57	Attempt to release communications buffer pool with ER LCORE$.
	61	Bad packet on ER SNAP$.
	62	A0 was negative on SETBP$ request.
	63	Relative breakpoint address outside user's program limits on SETBP$ request.
	64	"P" bit set in conjunction with "R" or "W" on SETBP$ request.
	65	Bit 34 set on breakpoint parameter for SETBP$ request.
	66	Attempt to load a segment into a nonbased dynamic bank, or the program control bank is a nonbased dynamic bank.
	67	An ACW in the absolute element is wholly or partially outside of the bank limits.
	70	Bad LIJ/LDJ. No bank currently assigned with index specified by LIJ/LDJ instruction or, captured P not equal to collector defined entrance to a guaranteed entry bank.
	77	Processor requested on ADED$ call is not available.
CONS (05)	0	Packet not within limits.
	1	Output buffer not within limits.
	2	Expected input count exceeds 50 characters.
	3	Input buffer not within limits.
REP (10)	1	RLIST$ packet not within program limits.
	2	LINK$ or RLIST$ request and REP's entry point is zero.
	3	RLIST$ request and either the file is not assigned or not on mass storage.
	4	RLIST$ entry name not found.
	6	REP contains D bank addresses.
	10	Bank Description Table (BDT) and extensions (BDTE) exceed system maximum (511 words) after RLIST$ request.
	11	RLIST$ request to remove previous REP list with REPs active.
	12	LINK$ or RLINK$ request and specified name not found by system's search.
	14	EXLNK$ or UNLNK$ request not from linked routine.
	15	Number of RLIST$ REP names exceeds system's maximum.
	16	A LINK$, RLINK$, or RLIST$ request and no PCT space available.
	17	A LINK$, RLINK$, or EXLNK$ request and system detected an I/O error in loading a REP. Or an I/O error reloading the REP after its storage had been released for time-sharing.
	20	The main program plus the REP's main storage requirements exceed total user main storage.

CONTROL STREAM DIAGNOSTIC MESSAGES

The following messages are among the most common and typical of the many hundreds in the system. A large number of other messages are worded somewhat differently, but have meanings which are similar to these.

When a code from 1_8 to 37_8 is contained in an error message, it often points to one of the I/O problems described under type 1, ERR MODE (EMODE) AND I/O STATUS CODES. Note that most of the messages issued by the FURPUR processor correspond to a specific I/O error and status code.

When a twelve-octal-digit status code is given in an error message, it often has bit settings corresponding to one or more of the causes of facilities rejection (FAC REJECT) or facilities warning (FAC WARNING) described at the end of this section.

Some diagnostic messages refer to operator keyins. Here are the usual meanings of the most common keyins:

A Try again with standard recovery.

B Return I/O status 12 to packet.

D Declare device down.

E Treat as end of file, or error off a run.

G Treat as unrecoverable error, since I/O device positioning appears to be good.

I Initiate a locked out or suspended symbiont.

L Lock out a symbiont.

N The reply is "no."

Q Re-enter a symbiont file in its appropriate queue.

R Reprint or repunch a symbiont file.

S Suspend a symbiont.

T Terminate a symbiont.

X Abort a symbiont, or abort a run.

Y The reply is "yes."

One of three abbreviations, SI (source input), RO (relocatable or absolute output), or SO (source output), is frequently used to identify the element named in the corresponding specifications subfield of a processor call statement, such as @ASM, @COB, @FOR, or @MAP. For processors such as @ELT which have no RO subfield, only SI and SO are meaningful.

■ DIAGNOSTIC MESSAGES

The self-explanatory messages are not included.

The run stream diagnostic messages are:

–@@COMPLETE

Normal completion of processing on a transparent control statement has taken place.

–@@ERROR (error codes)
–@@ERROR – ILLEGAL TYPE
–@@ERROR – SYNTAX

These three messages indicate that an error was encountered while processing a transparent control statement. The error codes are the same codes returned on a dynamic request of the control statement.

AT LEAST 1 PRINT IMAGE HAS BEEN TRUNCATED

An image length error occurred during file output and the attempted recovery was successful.

AWAIT/DEACT AMBIGUITY

The executive determined that all activities of the run were either in an AWAIT state (ER AWAIT$), a DEACT state (ER DACT$), an II$ wait state, or a Test and Set Queuing wait state and could not be activated by the run.

BAD INPUT SEQUENCE

The input control message was rejected because it conflicts with the current status of the run.

BRKPT

User's breakpoint setting matched and no contingency was registered.

CANNOT SYM PU$ WHILE IN USER FILE

Caused by @SYM PUNCH$,,CP control statement when the current PUNCH$ file has been @BRKPT'd to a user file.

CONTROL REGISTER VIOLATION (1110 Only)

The user attempted to reference EXEC GRS.

x CYCLE SPECIFICATION IGNORED

RO or SO cycle specification is meaningless and is ignored. Does not cause error return.

DATA IGNORED – IN CONTROL MODE

Data statements were encountered when the EXEC was attempting to read control statements; that is, a program or processor was not in control of the run at the time these statements were encountered.

DBANK CANNOT BE LOADED WITH NEGATIVE BD
DBANK RELATIVE STARTING ADDRESS = 0*sss*000, DBANK SIZE = 0*nnn*000, LAST ADDRESS OF USER CORE = 0*xxx*777.

The program cannot be loaded without causing BD to become negative. This is the result of using a SETMIN directive to specify a minimum starting address for the DBANK. See *UNIVAC 1108 Processor and Storage Programmer Reference, UP-4053* (current version), or *UNIVAC 1110 System Processor and Storage Programmer Reference, UP-7970.1* (current version).

ELEMENT UNOBTAINABLE *xx*

The element specified on the @START control statement cannot be found. *xx* is a program file search code.

@END IGNORED – IN CONTROL MODE

An @END control statement was encountered when the EXEC was attempting to read control statements; that is, the DATA OR ELT,D processor was not in control of the run at the time this statement was encountered.

@EOF IGNORED – IN CONTROL MODE

An @EOF control statement was encountered when the EXEC was attempting to read control statements; that is, a program or processor was not in control of the run at the time this statement was encountered.

EQUIPMENT TYPE ERROR, ADD FILE

ADD file was not in FASTRAND format.

ERROR – DYNAMIC DUMPS NOT CLOSED

I/O error encountered when writing system diagnostic file (DIAG$).

ERROR ENCOUNTERED WHILE OUTPUTTING FILE

An I/O error occurred with a code of 4_8, 11_8, or 12_8 and the inhibit recovery flag had not been set by an ER PRTCN$. The message is displayed if the recovery process is successful or if the operator did not terminate the file.

ERROR LOADING PROGRAM

An error was detected in the absolute element. Bad element or the element is possibly destroyed.

FAC REJECTED *xxxxxxxxxxx*

This message appears for a run that aborted due to a facility control statement that cannot be honored by the system.

FAC WARNING *xxxxxxxxxxx*

This message is a warning that the facility control statement could cause a problem.

FILE ALREADY IN USE

@BRKPT control statement issued for a file currently being used as a symbiont file not capable of being @BRKPT'd (e.g., an @ADD file).

x FILE CANNOT BE READ

Input file is in read inhibited mode due to absence of read key, write-only mode set for file, or Y option used on the file assignment.

FILE ERROR

The file requested on a @XQT or processor control statement could not be assigned or is not a program file. If the run is not demand, it is terminated.

FILE FORMAT ERROR. FILE TERMINATED

An image length error occurred and the I function was set.

FILE FORMAT ERROR. TERMINATED BY OPERATOR

An image length error occurred and the operator did not try recovery.

x FILE NOT FOUND – STATUS: *n*

File *x* is neither assigned to the run nor catalogued. *n* is the status returned when an attempt was made to assign file *x*.

FILE STMT ** FACILITY REJECT ** *xxxxxxxxxxxx*

If for some reason the processing of the @FILE statement by facilities was in error, this message is given, where the 12-digit code is the facilities status word.

FILE STMT * FORMAT ERROR*

The format of the @FILE statement is not consistent with predefined format specifications.

FILE UNOBTAINABLE *xxxxxxxxxxxx*

The file specified on the @START control statement cannot be accessed by the executive. *xxxxxxxxxxxx* is a 12-digit octal status code returned when the file cannot be assigned.

FIRST FILE NAME IS IN ERROR

First file name was not given for @BRKPT or @SYM control statement, or a @BRKPT control statement was for an inactive alternate file.

This message will also be printed if the D option is specified on a @SYM control statement and the filename specified is not the generic name PRINT$.

nn ILLEGAL CHARACTER *x*

The coarse scheduler encountered an illegal character x at column nn of the above control statement.

ILLEGAL COMMON BANK BDI *iii*

A common bank BDI number *iii* referenced by the program is invalid.

ILLEGAL CONTINUATION

Continuation of the above control statement is not allowed or the next control statement has the control character (@) in the first column. The control statement is not honored and the run is terminated (if it is not a demand run).

ILLEGAL ENTRY POINT TO A GUARANTEED ENTRY COMMON BANK

The entry point specified by the user program is not that of the guaranteed entry common bank which is initially based; or more than one guaranteed entry common bank is initially based.

ILLEGAL EQUIP ON @FILE

A file was assigned of the wrong device type. The user's read file is closed with a @FIN control statement, the run is marked as removed and the file from the first @FILE control statement is not catalogued.

** ILLEGAL EQUIP TYPE ON FIRST FILE **

The device type on the first @FILE control statement did not specify tape equipment. This may be encountered when processing multiple @FILE control statements, or when a 'T' option is found on the @FILE control statement for a mass storage file.

x ILLEGAL DEVICE

Output file is not FASTRAND format or input file is neither tape nor FASTRAND format.

x ILLEGAL FIELD

Field is ambiguous with option given (for example, I option specified and source output field coded).

ILLEGAL LIJ/LDJ INSTRUCTION
STATUS WORD = *xxxxxx*

On an LIJ or LDJ instruction, the user had either an E-bit violation or table length violation. *xxxxxx* of the status word will be bits 35–18 of the undefined sequence interrupt status word.

nn ILLEGAL OPTION *x*

An illegal option *x* was encountered at column *nn* of the above control statement. The control statement is not honored and the run is terminated (if it is not a demand run).

IMAGE IGNORED – TRANSP CTL IN PROGRESS

A transparent control statement has been rejected because another transparent control statement from this terminal is currently being processed.

IMPROPER RUN STREAM IN FILE

The first image in the file or element specified on the @START control statement is an invalid @RUN control statement.

INVALID SYMBIONT NAME

@SYM file is directed to an illegal or non-existent device or group.

I/O ERROR ENCOUNTERED

The executive returns this message after receiving an error code while trying to read the file specified on the @START control statement.

I/O ERROR *xx*. FILE TERMINATED

An I/O error occurred which is not one of the recoverable types (4_8, 11_8, and 12_8).

I/O ERROR IN TERMINATION – PMD NOT INITIALIZED

I/O error encountered when writing system diagnostic file (DIAG$).

I/O ERROR *xx*. TERMINATED BY OPERATOR

An I/O error occurred and the operator terminated the file. xx is the error code.

I/U OPTION CONFLICT

Both I and U options given on processor control statement – ambiguous options.

LABEL FORMAT ERROR

User tried to illegally access a labeled tape or hardware error occurred when trying to validate a tape/disc pack label. Also, operator responded E to a request to mount a disc pack or tape.

LAST REL I–BANK ADDR GTR '0177777'

The hardware field 'BS' in the PSR has 7 significant bits. This field is meaningless if set above 0177777.

L IS LEGAL FOR TAPE ONLY

The L option was specified on the @BRKPT control statement and file does not reside on tape.

MASS STORAGE OVERFLOW

Mass storage request cannot be satisfied because mass storage is not currently available.

nn MAX NUMBER OF CHARACTERS EXCEEDED

The character at column *nn* of the above control statement is not a field/subfield terminator and the maximum number of characters for this subfield has been reached. The control statement is not honored and the run is terminated (if it is not a demand run).

nn MAX NUMBER OF FIELDS OR SUBFIELDS EXCEEDED

The character at column *nn* of the above control statement is the field terminator and no more fields are allowed for the control statement, or the character is a subfield terminator and no more subfields are permitted for that particular field. The control statement is not honored and the run is terminated (if it is not a demand run).

NO FILE SPECIFIED

File name is not specified on a @START control statement.

NO RUN ACTIVE

Applies only to demand processing – message appears on demand terminal if statements are entered before the @RUN control statement.

NO SPACE FOR MAJOR SAVE ON ABORT$ CONTINGENCY

No space available in PCT for register save when processing an ABORT$ contingency. Run is terminated.

NON-ZERO I/O STATUS FROM USER FILE *ss*

A bad I/O status was returned after an I/O request on the user's file. The file has been destroyed or there is incorrect data in the absolute element.

OPERATION IS ILLEGAL FOR DEMAND

@BRKPT PRINT$ or @BRKPT PRINT$/PRINT$ control statement from a demand terminal before @BRKPT PRINT$/file.

filename OUTPUT FILE IS TAPE

Output file should be FASTRAND format and is tape instead.

*** PARITY ERROR ***

A parity error has been detected in at least one character of the input image. The entire image is discarded. (For teletypewriter only.)

PCT EXPANDED BEYOND SYSTEM LIMITS

The number of main storage blocks required for expansion of this run's PCT exceeds the system generation parameter PCTMAX. When a run aborts with this message, a postmortem dump of the PCT (obtained using @PMD,P) may show one of the following to be the cause:

(1) Excessive number of granule tables (change track granularity to position granularity)

(2) Excessive number of activities (check for ER FORK$ loop)

(3) Excessive number of files assigned (check for ER CSF$ loop)

PCT OVERFLOW ON INITIAL LOAD

The total number of PCT blocks requested is greater than the system's generation parameter PCTMAX. This message is specifically given during an initial load.

PCT/PROGRAM SIZE EXCEEDS USER CORE

Either an internal main storage bank has been downed so the program does not fit or a real-time program has started and this program is too large to fit the available main storage.

PMD NOT ALLOWED

Postmortem dump is not allowed for a system processor (called from the SYS$*LIB$) unless a Y option appeared on the @RUN control statement. If an N option appeared on the @RUN control statement, no postmortem dumps of any programs are allowed.

PRIVILEGED INSTRUCTION VIOLATION (1110 only)

The user attempted to execute an EXEC-only instruction.

PROGRAM NOT FOUND

The requested program or processor is not in the given file, LIB$, or TPF$ (depending on the statement). If the run is not demand, it is terminated.

PROGRAM TOO LARGE PROGRAM SIZE *nnn* BLKS.
CORE SIZE *xxx* BLOCKS.

The program is too large to fit in the space available to user programs. The program requires *nnn* main storage blocks and user main storage consists of *xxx* main storage

blocks. A main storage block is 512_{10} (1000_8) words and the sizes *nnn* and *xxx* are octal numbers, thus giving the sizes in octal 1000's.

x READ ONLY OUTPUT FILE

Output file is in write inhibited mode, due to absence of write key, read-only mode set for file, or Y option used on the file assignments.

REAL TIME PROGRAM ATTEMPTED PCT EXPANSION

A PCT expansion attempt for a real-time program could not be done. If the program expects PCT expansion it should use a @RUN control statement option to initialize PCT size so expansion is not attempted.

nn REQUIRED FIELD OR SUBFIELD MISSING

A field or subfield which is required on the above control statement has not been specified. The omission was detected when the field/subfield terminator or the end of the control statement was encountered at column *nn*. The control statement is not honored and the run is terminated (if it is not a demand run).

RUNSTREAM ANALYSIS TERMINATED

The run has been terminated because of an error condition and the remaining control statements are not processed.

SECOND FILE NAME IS IN ERROR

Caused by second file name on @BRKPT control statement not currently assigned to the user.

SECOND @FILE STATEMENT FORMAT ERROR

A syntax error was encountered on the second @FILE control statement. SECOND here implies that one or more @FILE control statements have already been processed. The user's read file is closed with a @FIN control statement, the run is marked as removed and the file from the first @FILE control statement is not catalogued.

SECOND NAME IS ILLEGAL

Second file cannot be given on a @BRKPT of a read alternate, print alternate, or punch alternate file.

SI: CYCLE NON-EXISTENT OR IN ERROR

Requested cycle of specified element does not exist or cycle field has improper format.

SI: ELEMENT NOT FOUND

Element name given cannot be found as a symbolic element in the specified program file.

SI: IMPROPER LABEL BLOCK

Source input file is tape, and tape is not positioned at the label block for requested element, probably because a @FIND has not been done.

SI: MISSING FIELD

A field of required information (for example, element name) was not given.

SIRASM EDIT ERR *c r e*

Line correction diagnostics produced by SIRASM in the edit mode.

where:

- *c* — Indicates the cause of the error. A list of possible causes is shown below.
- *r* — First four words of the range correction statement under whose control the error occurred.
- *e* — Specifies the change correction statement that caused the error.
  - □ SEPARATOR — The separator used in the change correction statement is invalid or nonexistent.
  - □ COLUMN — The column number specified on a format 1 or 2 change correction statement is out of range, or that C > D for a format 2 change correction statement.

- NO FIND — The characters given in the old data parameter of a format 3 or 4 change correction statement could not be found in the line being corrected.

NOTE:

Whenever one of the above errors occurs, the change correction statement is ignored and the line remains unchanged.

- ASCII MODE — Indicates that symbolic input or output is in ASCII code, or that the user requested ASCII code. Since SIRASM cannot correct ASCII code, all range and change correction statements are ignored.
- CARD COUNT <– Not enough change correction statements were provided. Those lines for which no change correction statement was provided remain unchanged.
- CARD COUNT >– Too many change correction statements were provided. The excess change correction statements were ignored.

STORAGE LIMITS/WRITE PROTECT VIOLATION (1110 only)

The user attempted to reference an area outside his limits or attempted to write into a read-only area.

TAPE IMAGE LOST*REREAD

Applies only to demand terminals – indicates that images were lost while inputting images in form II paper tape mode.

TAPE OR TEMPORARY FILE NOT ALLOWED

The user attempted to perform a @START of a run stream on a tape file or temporary file.

TIME OUT WARNING

No activity has occurred on the line for a predefined interval. If another time interval elapses without activity, the terminal is terminated.

TIMEOUT ON INTERRUPT LOCKOUT (1110 only)

The user was taken to guard mode interrupt because interrupts were locked out for more than 100 usec.

TOO MANY COMMON BANKS DEFINED

The number of common banks referenced by this program is greater than 1500 decimal.

TOO MANY USER BANKS DEFINED

The number of user banks defined for this program is larger than the maximum allowed, 250 decimal.

UNRECOVERABLE I/O ERROR WHEN READING FILE *filename*

The coarse scheduler encountered an unrecoverable I/O error when searching file *filename* for a program or processor. If the run is not demand, it is terminated.

WAIT LAST INPUT IGNORED

Applies only to demand terminals – indicates that the system is not ready for further input.

WARNING IMPROPER OPTION

Caused by @SYM,C control statement with printer symbiont name (warning only).

FACILITY REQUEST STATUS CODES

If a facilities request made by one of the facilities control statements (@ASG, @MODE, @CAT, @FREE, and @USE) is found to be in error, a status word is generated in which the various bits set define the error. For incorrect facilities control statements submitted in the run stream, the status word is given as part of the FAC REJECTED . . . or FAC WARNING . . . message. For control statements submitted by a CSF$ request, the status word is returned in register A0.

Bit Set	Description
35*	Request not accepted; check other bits for reason.
34*	Field error in control statement other than syntax. Option conflict (MHL, OE, or IB) or noise constant specification error. Requested hardware not currently part of the system.
33	File is already assigned for @ASG or @CAT control statement specified, already freed for the @FREE control statement specified, or not assigned for the @MODE control statement specified. This setting is fatal for @CAT and @MODE control statements.
32	The file was previously catalogued.
31*	Equipment type specified on @ASG control statement is not compatible with catalogued type or file specified on @MODE control statement is not magnetic tape.
30	Not used.
29	The portion of the filename used as the internal name for I/O packets is not unique.
28	Not used.
27*†	Incorrect read key for catalogued file.
26*†	Incorrect write key for catalogued file.
25	Write key that exists in the master file directory is not specified in the @ASG control statement (file assigned in the read-only mode).
24	Read key that exists in the master file directory is not specified in the @ASG control statement (file assigned in the write-only mode).
23*†	Read key specified in the @ASG control statement; none exists in the master file directory.
22*†	Write key specified in the @ASG control statement; none exists in the master file directory.
21*	An 'A' option was specified in the @ASG control statement and the filename cannot be found in the master file directory.
20*	Invalid reel number specified in the @ASG control statement for a catalogued tape file, or pack-id for catalogued removable disc file.
19*	Mass storage file has been rolled out.
18*	Request on wait status for facilities. For a tape file, this usually means a tape unit is not currently available. For a drum file, this usually is caused by an exclusive use conflict with another concurrent run.
17*	Option conflict for catalogued files, either the D and K options were specified or C or U, or P, R or W in combination with C or U was specified for a file which already exists in the directory.
16*	File assigned exclusively to another run.
15	Find was made on a catalogued file request and the file was already assigned to another run.
14*	File to be decatalogued when no run has file assigned.
13*	Project-id incorrect for catalogued private file.
12	Equipment is tape.
11	Read-only file catalogued with an R option.
10	Write-only file catalogued with a W option.
9	Equipment requested is down.
8*	File specified in an @ASG control statement is disabled because the links pertinent to its master file directory items have been destroyed.
7	File specified in an @ASG control statement has been disabled because the file was assigned write-enabled during a file recovery.

Bit Set	Description
6*	File specified in an @ASG control statement has been disabled because the file has been rolled out and the backup copy is unrecoverable.
5*	F-cycle conflict (a) Cataloguing of the requested F-cycle would force deletion of a currently assigned F-cycle. (b) F-cycle generation inhibited due to existence of +1 file. (c) F-cycle requested is not within the currently acceptable range.
4–0	Reserved for future use.

*Request was rejected. If request was submitted by the run stream, this results in a FAC REJECTED message and termination if a batch run (results in only a FAC REJECTED message for demand runs). For dynamic requests through a CSF$ request, bit 35 is set in the status word returned in register A0.

†If the statement was submitted by a CSF$ request, the run is aborted and no status word is returned in register A0.

5. DEMAND PROCESSING

The executive supports the use of the following terminals to access the system in the demand mode:

(1) UNISCOPE 100 Display Terminal

(2) UNISCOPE 300 Visual Communications Terminal

(3) UNIVAC DCT 1000 Data Communications Terminal

(4) UNIVAC DCT 500 Data Communications Terminal (Teletypewriter and Semi-automatic Mode)

(5) Teletype Models 33, 35, 37, and 38 (KSR and ASR)

(6) Friden Model 7100 Typewriter

GENERAL DEMAND TERMINAL OPERATIONAL PROCEDURES

■ INITIALIZATION

Before the demand terminal can be initialized, the user must turn it on, set the various switches to the proper position, and establish the proper line connection if operation is on a switched line network.

Once the connection is made and the terminal is initialized, the demand user must send a six-character remote site-id to the operating system.

The site-id submitted by the demand user is compared to a list of valid site-ids and if the system responds with the message:

UNIVAC 1100 TIME-SHARING EXEC VER xx.xx.xx

The demand user can assume that the initialization operation is completed (xx.xx.xx is the version of the UNIVAC 1100 Series Operating System operational at the central site).

■ DEMAND TERMINAL MODES OF OPERATION

The demand terminal has three distinct modes of operation:

(1) Terminal Inactive Mode

This is the initial mode of the terminal following the sign-on procedure. The terminal will return to this mode at the completion of the other two modes.

(2) Demand Run Mode

This mode is achieved by submitting a @RUN control statement from the primary input device, that is, the keyboard. In demand mode, the input will be solicited when input is desired by the executive. The terminal is returned to the inactive mode by submitting a @FIN control statement. Another @RUN control statement will not be accepted while in the demand run mode.

(3) Remote Batch Mode

The demand terminal may be switched from the terminal inactive mode to the remote batch mode for input or output.

The 'B' option on the @RUN control statement (@RUN,/B) will place the terminal in the remote batch input mode. Input will not be solicited as in demand run mode. The terminal will be returned to the terminal inactive mode following a @FIN control statement. Another @RUN control statement will be accepted while in the remote batch input mode and will be treated as another remote batch run whether it contains a 'B' option or not. Output files generated by the remote batch run as well as those SYM'd (via @SYM) to the terminal can be displayed at the terminal by entering the statement, @@SEND. This mode can be achieved only from the terminal inactive mode. The terminal is returned to the terminal inactive mode when the output process of the file is complete.

■ DEMAND SYMBIONT CONTROL STATEMENTS

Control of demand symbionts is regulated by control statements prefixed with a double master space (@@).

These control statements do not require the input solicitation. They may be entered after an output interrupt (break-key) or any other time the terminal operator finds the need. Each demand symbiont may have control statements for its individual features.

Statement	Mode	Description
@@X TIOC		The @@X statement directs the executive to take action on any or all of the four possible action parameters.
	Demand Run	T – terminate the demand run's present execution.
	Demand Run	I – discard all backed-up input.
	Demand Run Remote Batch	O – discard all backed-up output.
	Demand Run	C – generate a 'BRK' contingency.
@@SKIP n	Demand Run Remote Batch	Skip n lines of output where n is a value of 0 to 63. The SKIP may be reset by a @@SKIP 0.
@@SEND	Terminal Inactive	Send any queued batch output file to the terminal.
@@RQUE	Remote Batch	Stop the present batch output file and requeue it for a later @@SEND request. Return to terminal inactive mode.
@@CONT	All	Directs the symbiont to continue. Useful after a BRK-KEY when no action is desired.
@@CQUE	Demand Run	Circumvent input solicitation requirement. Allow several input images to be buffered in memory before the terminal is placed in the wait condition.
@@INQ	All	Directs the executive to buffer all input to mass storage until the @@END control statement is received. If the @@INQ statement is entered in terminal inactive mode, the next input should of course be a @RUN statement. All run statements entered while in @@INQ mode will be considered remote batch and not demand.
@@END	All	Terminates special input mode, i.e., @@CQUE or @@INQ. The @@END returns the terminal to demand run from @@CQUE and will begin processing the mass storage buffered input for @@INQ.
@@ESC	Demand Run	Allows the input to be passed to the requestor unaltered from the format of which it was entered; that is, all communication envelope characters are not removed nor is the image translated.
@@TERM	All	Directs the executive to terminate the terminal. It is recommended that the remote operator enter @@TERM only while in terminal inactive mode. However, if entered while a run is active, the run and terminal will be terminated.

■ DEMAND TERMINAL TERMINATION

The standard termination procedure is performed when a @FIN control statement is received by the system. The executive retains control of the line terminal until all output destined for the site has been processed.

The symbiont then returns to the system command mode. Either another @RUN control statement or the termination sequence, @@TERM should follow.

TELETYPEWRITER/DCT 500 SYMBIONT

The TTY/DCT 500 symbiont provides support for Teletype Models 33, 35, 37 and 38 (KSR/ASR), Friden* Model 7100, and the DCT 500 operating in teletypewriter or semi-automatic mode.

**Trademark of Friden Division of Singer Company*

■ PAPER TAPE OUTPUT OPERATIONS

No special effort is required to force output to paper tape on TTY mode devices. All that must be done is to turn on the paper tape punch and output will occur on both the printer and the punch. However, the semiautomatic DCT 500 processes output to the punch or printer (not both) and command statements are provided to specify which device is to be used.

Statement	Description
@@PTO	DIRECT OUTPUT TO PAPER TAPE (for the DCT 500 in semi-automatic mode)
@@PRO	DIRECT OUTPUT TO PRINTER (for the DCT 500 in semiautomatic mode)

If the paper tape punch cannot be accessed due to some hardware problem, the following message will be displayed:

PAPER TAPE PUNCH COULD NOT BE SELECTED

■ PAPER TAPE INPUT

Paper tape input is supported for tapes having the format data -CR-LF -data -CR-LF. The LF is needed only for readability at the terminal and may be deleted. The following procedure governs the use of paper tape input:

(1) The paper tape reader should be prepared for input.

(2) The user must enter the statement,

@@PTI

(3) The system will respond with the message:

START PAPER TAPE INPUT

(4) The paper tape will be read.

(5) To terminate paper tape input, enter CNTRL-S (X-OFF or DC3) from the keyboard or from the tape (the latter is desirable). Multidropped semiautomatic DCT 500s will terminate tape input mode when the paper tape is completed. It is, however, desirable to have the CNTRL-S punched into the tape.

When paper tape input is completed, the symbiont will respond with this message:

END PAPER TAPE INPUT

(6) Paper tape input will be terminated without operator intervention if a parity error is discovered or if data loss occurs.

(7) If a real-time program prevents the symbiont from properly servicing paper tape input and data is lost, the following message will be displayed:

PAPER TAPE INPUT DATA LOST

The following are special considerations for the utilization of paper tape input:

(1) A complete run stream may be entered on paper tape. The executive will treat this as a batch run. This must be done when the terminal is in inactive mode.

(2) A @RUN without a @FIN may be entered via paper tape. The run will be treated as batch input. Subsequent input from the keyboard will be treated as a continuation of the batch input.

(3) Paper tape input may contain nongraphic characters (which are normally illegal) if the terminal is in input escape mode.

■ SPECIAL CHARACTERS

The TTY/DCT 500 symbiont accepts several ASCII input characters as having special meaning. The interpretation of these characters may be changed dynamically by the user via the @@TTY and @@DCT symbiont control statements.

ASCII Character	Keyboard Position	Function	Description
CAN	CNTRL-X	Line Delete	The current image is discarded. The symbiont responds with a CR/LF sequence.
SUB	CNTRL-Z	Character Delete	One preceding character is deleted each time the SUB character is sent.
CR	RETURN or CR	End of Image	Used to indicate the end of an input image.

The output character > is used to denote input solicitation, i.e., the user may input.

- **INTERRUPTING OUTPUT PROCESSING**

The break key is represented on the keyboard as BREAK or RTS or INT or INTERRUPT. It is used to temporarily terminate output so that the user may enter a demand symbiont control statement or a transparent (system) control statement.

Upon receiving the break/interrupt, the symbiont will print the following message:

OUTPUT INTERRUPT

- **OPERATION MODIFICATION CONTROL STATEMENTS (@@TTY,@@DCT)**

The TTY/DCT 500 symbiont provides the @@TTY and @@DCT symbiont control statements which allow the user to change certain parameters associated with his terminal. The @@TTY and @@DCT control statements are interchangeable (i.e., may be submitted from either device) and all parameters on the control statement are optional.

Format:

@@TTY char-1, value-1, ..., char-n, value-n
or
@@DCT

The character is a single alpha character selected from:

T Image terminator
L Line delete
C Character delete
S Solicit
W Page width
P Page length

The value may either be a character (with the exception of page width or page length) or an octal or decimal value. If a single digit is used, this is treated as a character.

DCT 500 IN TELETYPEWRITER MODE

Once the terminal has established a line connection with the central site, the terminal operator must depress the PROCEED key to establish clear-to-send at the DCT 500. The CLEAR TO SEND indicator lights, if the data set is in data mode when the PROCEED key is pressed. Once this sequence is performed, the terminal operator can send a site-id to the system.

The site-id must be submitted with upper case alphabetics. The second character of the site-id must be a D to signify that the terminal is a DCT 500.

- **SEMIAUTOMATIC DCT 500**

A brief description of the switches, indicators, and general operation of the device is given here to facilitate demand use of the DCT 500. It should be noted that not all DCT 500 configurations have all these switches, and thus, not all the capabilities.

Switch	Description
MASTER/SLAVE	Permits the DCT 500 to be the initiating station if in MASTER position. To operate with the symbiont, this switch must be in the SLAVE position.
XMIT OFF/RECMON	This switch is effective only when the DCT 500 is in full-duplex mode. Since the DCT 500 must be in half-duplex mode to operate with the symbiont, the setting of this switch has no function.
BAUD RATE	This switch is used to set the clock in the DCT 500 to the same rate as the clock in the communications terminal at the central site to which the DCT 500 is connected. The operator must know the rate of the line he is using and set this switch accordingly.
ON LINE/OFF LINE	This switch must be set to ON LINE in order to make connection with the computer.
KEYB'D/OFF	Since the primary input device is the keyboard, this switch must be in the KEY'BD position.
PRINTER/OFF	This switch should be in the PRINTER position to allow the printer to be selected.
READER/OFF	This switch should be in the OFF position if paper tape is not to be read. It must be in the READER position before paper tape can be read.
PUNCH/OFF	This switch should be in the OFF position if paper tape is not to be punched. It must be in the PUNCH position before paper tape can be punched.

After the line connection is made, the terminal operator should watch the CLEAR TO SEND indicator to know when he has been polled and is able to enter the site-id for his terminal. Any letters in the site-id must be entered in upper case by striking the SHIFT key along with the letter key. Two time restrictions are placed on the operator in the entering of the site-id. He has only a few seconds from the time the CLEAR TO SEND indicator lights to enter a character. If this time passes without a character being sent, the indicator extinguishes indicating he has been deselected and he must then wait to be polled again. After the first character has been struck, he has 15 seconds to enter a valid site-id. The terminal is deselected if this time elapses and the operator must again wait to be polled again. When the valid site-id is received, the standard initialization message is sent to the terminal printer. The terminal is now considered an active terminal, and the keyboard is selected so that the user can start his input.

UNISCOPE 100/DCT 1000 SYMBIONT

■ OPERATIONAL CONSIDERATIONS FOR THE UNISCOPE 100

(1) Before the UNISCOPE 100 becomes an active terminal, the operator must turn it on and establish the proper line connection. The first message transmitted from an inactive device must be the six-character site-id for this device. The site-id may be transmitted from anywhere on the screen, preceded by an SOE (▷) character.

If the message is not received properly, no response is sent to the device. If the operator is sure that everything is functioning properly and that the transmitted site-id is correct, then the following steps should be taken after a waiting period of about 60 seconds:

(a) Press the wait switch (to unlock the keyboard).

(b) Transmit the message again.

The operator sees a positive action when the device is polled to pick up the transmitted message by the reappearance of the cursor character. The cursor disappears from the screen when the transmit key is pressed and reappears when the device is polled.

(2) Input sent to the computer consists of the data between the cursor and the previous SOE or between the cursor and the screen origin, if no SOE precedes the cursor. This input is broken into images, each image occupying at most one line of the screen. Unless full-screen input has been enabled via the @@FUL statement, multiple images will not be accepted. All will be rejected if more than one is sent.

After the operator has pressed the transmit key, the cursor disappears from the screen. If the message is received properly by the computer, any output is sent to the insert point. The symbiont generates and sends a keyboard unlock message, moves the screen up or down (depending on the current setting) one line, and positions an SOE character and the cursor in columns one and two of the insert line. The operator is free to enter a new line to be transmitted when the SOE appears.

(3) The symbiont controls the screen with output messages.

The operator may interrupt output via the message waiting key. The computer will acknowledge the interruption, and the operator is then free to enter any input, specifically including demand symbiont control statements.

(4) It is the terminal operator's responsibility to remove his device from the active status when he is finished. This is done by transmitting the demand symbiont control message @@TERM. If he does not, the terminal will timeout.

■ OPERATIONAL CONSIDERATIONS FOR THE DCT 1000

Initialization Procedure:

(1) Set switches to the following positions:

AUTO/MAN	– AUTO
MONITOR ON/OFF	– ON
ON LINE/OFF LINE	– ON LINE
KEYB'D/OFF	– KEYB'D
ALL OTHER DEVICE SWITCHES	– OFF

(2) Establish line connection.

(3) Press clear key and set RUN/STOP switch to stop position and then run.

(4) Enter the six-character site-id for this terminal from the keyboard.

If the transmitted site-id is valid for this particular terminal and is received properly, the standard message is printed at the terminal.

All demand runs submitted via the DCT 1000 must have their @RUN control statements entered via keyboard. All @RUN control statements entered via cards or the paper tape reader initiate batch runs.

■ OPERATOR SCREEN AND INPUT/OUTPUT CONTROL STATEMENTS

The following may be requested for the UNISCOPE 100:

Statement	Description
@@RLU	Set screen roll direction to up. There is no response to this message.
@@RLD	Set screen roll direction to down. There is no response to this message.
@@FUL	Enable full-screen input (queued mode). In this mode, up to full screens of data will be accepted as multiple images. Output will not be transmitted, but will be queued and sent when normal input mode is reinstated. There are two possible responses: one indicates that the request has been accepted, the other that facilities are not available at this time for queued mode.
@@INS xx	Set screen insert point at xx, where xx is number of lines on screen. If xx is omitted, the original system setting specified in the system's generation is used.
@@PRNT xx	Start printer on COP number xx. An error message is returned if the COP is not configured or cannot be selected. Operands are assigned in the system's generation and are mandatory, as the @@PRNT will be ignored without one. Up to six COPs can be configured for each UNISCOPE 100.
@@NOPR	Stop printer on COP.

COP is the UNIVAC Communications Output Printer that is used to obtain a hard copy listing of the UNISCOPE 100 screen.

Requests valid for either DCT 1000 or UNISCOPE 100 terminals are:

Statement	Description
@@ESCA	Enable escape mode for input. Cursor positions will not be stripped from the input data while in escape mode. For example, if a terminal operator were to @XQT program, and then @@ESCA, the program could determine the cursor positions by examining the input data returned via an ER to AREAD$. On output, the user can keep the symbiont from providing cursor positioning by placing an ESC (033) as the first character of the image which is to be sent to the screen via an ER to APRINT$. If he does so, he must then provide any cursor control.
@@END	End special mode. Returns terminal from any of the special modes (escaped and queued) to normal input mode.

DCT 1000 Requests

Statement	Description
@@PRNT	Select printer for output.
@@PTP	Enter point-to-point mode. This mode must be configured or the command will be ignored.
@@PTI/@@CDI	Select queued mode and enable paper tape or card input. The DCT 1000 must be configured with a tape reader or card reader. One of two messages may be returned to the terminal indicating that either the request for queued mode was rejected or accepted. If queued mode is accepted, the operator must: □ Clear the KEYBOARD switch. □ Set the paper tape reader or card reader switch. □ Throw the RUN switch to STOP, then to RUN. The input will then begin. An @@END must be submitted after the input to allow the queued output to be sent (see @@ESCA).
@@PTO or @@CDO	Enable paper tape or card output. This message type allows for computer selection of the appropriate output device. A @@PRNT must be entered following all punching to redirect output to the printer. The operator may override the output device selected by the demand symbiont control statement. Unless he chooses to do so, no switch settings are necessary.

UNISCOPE 300 SYMBIONT

■ OPERATIONAL CONSIDERATIONS

(1) Since many UNISCOPE 300s lack @ or + keys, the symbiont translates # and &, respectively, into @ and +, if specified in the system's generation.

(2) After the operator has dialed the line with which his UNISCOPE 300 is configured, he initializes the station with the following procedure:

(a) Set up message on screen: SOM, (term-id), cursor

For example: ΔTSL001 (only enough characters (six) to contain the term-id will be read – any others will be discarded).

(b) Press the transmit key.

(c) Wait for the standard initialization message.

(d) Submit a @RUN control statement (using normal executive format, except that # may be used in place of @).

(3) Input images use the format:

(a) Start-of-message character (△).

(b) Text (in 64-character images, as many as the screen can hold).

(c) Cursor.

For example: △TEXT . . .

The input is initiated by the operator pressing the transmit key. This action also lights the wait indicator. Acceptance by the symbiont is indicated by the appearance of the SOM character and the extinguishing of the wait indicator.

■ **SPECIAL CONTROL SEQUENCES**

The message waiting key provides the ability to interrupt output to send input messages of control sequences to the central site. Output resumes when the transmit key is pressed. The message waiting key may be pressed at any time on a multistation line, but on a single station unit it must be pressed just after the output has been received and prior to the subsequent acknowledge poll. If the user's timing is wrong, the message waiting will be ignored, and he should keep trying until the central site acknowledges his attempt.

The demand symbiont control statements for the UNISCOPE 300 are the same as those for the UNISCOPE 100 with some exceptions discussed briefly below.

Statement	Description
@@FRZ nn	Freeze screen above line nn (two digits must be sent). The portion of the screen above line nn will no longer be rolled up or down.
@@INS nn	Same as UNISCOPE 100, except that two digits must be sent.
@@HI	Use fastest mode output (all scopes initially use this).
@@MED	Use medium rate output (about half speed of @@HI).
@@LOW	Use slow rate output (about one third of fastest).
@@PRNT xx	Assign PAGEWRITER xx. If the PAGEWRITER is already assigned to some other scope on the same MSCU unit, a message will be sent indicating the assignment was not accomplished. The terminal is put into @@LOW rate of output, regardless of whether the assignment could be done or not. If a PAGEWRITER is in use, and the symbiont cannot communicate with it, an error message will be sent.
@@NOPR	Release assigned PAGEWRITER.

TERMINAL SECURITY SYSTEM

■ **LOGGING-ON**

This section applies only if the executive terminal security system is configured into the system. Since many different sequences are available, only three primary methods are shown.

The asterisk (*) preceding the user-id takes on a special meaning. If the user-id is so configured, the asterisk informs the terminal security system that the user wishes to supply his own @RUN statement. If the asterisk is absent, the terminal security system will generate the @RUN statement for the user if so configured.

■ **THE BASIC METHOD (AN INSTALLATION OPTION)**

The user must enter a SITEID, a USERID (preceded by an asterisk), and a PASSWORD. The SITEID depends upon the terminal which is being used. The USERID and PASSWORD are assigned to the user by the site. In this example, the USERID is JONES and the PASSWORD is HAPPY.

When Jones logs on the system, the following occurs:

(1) Jones enters his SITEID:

SU1801

(2) The system responds with:

ENTER USERID/PASSWORD

(3) Jones must then enter

*JONES/HAPPY

followed by a TRANSMIT or carriage return.

(4) Jones is now logged on. He is now ready to enter a @RUN statement. The system will roll the USERID and PASSWORD off the screen for UNISCOPEs or space up 10 lines and print *DESTROY USERID/PASSWORD ENTRY* for printer terminals and, if the USERID is valid, respond with the system header.

■ THE RUN METHOD (AN INSTALLATION OPTION)

The run method is similar to the basic method. The run method causes a @RUN image to be generated internally. In this example, the USERID is SMITH and the PASSWORD is REDDOG.

Example:

```
SU1801              site-id
ENTER USERID/PASSWORD:
SMITH/REDDOG
UNIVAC 1100 OPERATING SYSTEM VERS. xx.xx.xx
DATE: 061272     TIME:  101242
RUN NUMBER: 000012
LAST RUN AT: 061172    093457
```

Note that the asterisk is not used in front of the USERID. Note also, that the system displays a run number (cumulative for the USERID), the date and time of the last run.

■ USE OF THE TERMINAL SECURITY SYSTEM PROCESSOR

The Terminal Security System (TSS) processor is provided to allow a user to influence his USERID/PASSWORD environment.

The TSS processor is invoked by the @TSS executive control statement. One purpose of TSS is to give each time-sharing user access to his own security information. This capability is dependent upon the individual site configuration and may not be available at every site or to every user at a site.

The TSS processor provides several commands for the user.

– PWORD COMMAND

The PWORD command is used to add a password to the list of passwords acceptable for the user specified by the USERID given during log-on. The format is:

PWORD new-password

The new password must be an alphanumeric of six or less characters.

– PLIST COMMAND

PLIST is used to obtain the list of legal passwords for the user. The command format is:

PLIST

The output format is:

PASSWORD: password-1
PASSWORD: password-2
etc.

– ALTER COMMAND

The ALTER command allows a user to change a password. The format is:

ALTER old-password, new password

– **EXIT COMMAND**

EXIT causes the TSS processor to terminate.

– **ERROR MESSAGES**

Several error messages are possible from the TSS processor.

UIFILE DISABLED	–	The security file has been destroyed and no updates are possible.
USER UPDATES NOT ALLOWED	–	The system does not allow the terminal user to use TSS.
COMMAND NOT RECOGNIZED	–	An invalid command was entered.

6. HARDWARE CHARACTERISTICS

ASCII AND FIELDATA CONVERSION TABLE

Codes, which also represent collating sequence, are given in octal.

ASCII codes from 00_8 to 37_8 are for communication, format, and separator control characters. These are not converted into Fieldata.

The ASCII symbols represented by codes 40_8 to 137_8 are converted into the identical Fieldata symbols, except that the quotation marks symbol is converted into a lozenge, the circumflex is converted into a delta, and the underscore is converted into a not equal sign.

There are no remaining unique Fieldata symbols into which to convert the balance of the ASCII symbols, represented by codes 140_8 to 177_8, so these codes are "folded" over codes 100_8 to 137_8 (by clearing bit 5, which amounts to subtracting 40_8). This means that ASCII codes 101_8 (A) and 141_8 (a), for example, are both translated as if they were code 101_8 (converted to Fieldata 06_8 for A).

Although ASCII is presently a seven-bit code, it may eventually be extended to eight bits to allow for additional controls and special graphic characters, including possibly whole alternate alphabets. On a 36-bit machine, each ASCII code is stored within a 9-bit quarter word.

FIELDATA TO ASCII CONVERSION

FIELDATA		ASCII		FIELDATA		ASCII	
OCTAL CODE	SYMBOL	OCTAL CODE	SYMBOL	OCTAL CODE	SYMBOL	OCTAL CODE	SYMBOL
00	@	100	@	40	)	51	)
01	[	133	[	41	–	55	–
02	]	135	]	42	+	53	+
03	#	43	#	43	<	74	<
04	Δ	136	^	44	=	75	=
05	SP	40	SP	45	>	76	>
06	A	101	A	46	&	46	&
07	B	102	B	47	$	44	$
10	C	103	C	50	*	52	*
11	D	104	D	51	(	50	(
12	E	105	E	52	%	45	%
13	F	106	F	53	:	72	:
14	G	107	G	54	?	77	?
15	H	110	H	55	!	41	!
16	I	111	I	56	'	54	'
17	J	112	J	57	\	134	\
20	K	113	K	60	0	60	0
21	L	114	L	61	1	61	1
22	M	115	M	62	2	62	2
23	N	116	N	63	3	63	3
24	O	117	O	64	4	64	4
25	P	120	P	65	5	65	5
26	Q	121	Q	66	6	66	6
27	R	122	R	67	7	67	7
30	S	123	S	70	8	70	8
31	T	124	T	71	9	71	9
32	U	125	U	72	´	47	´
33	V	126	V	73	;	73	;
34	W	127	W	74	/	57	/
35	X	130	X	75	.	56	.
36	Y	131	Y	76	¤	42	"
37	Z	132	Z	77	≠	137	—

FIELDATA OCTAL CODE TO CARD PUNCH CONVERSION

OCTAL CODE	SYMBOL	CARD CODE	OCTAL CODE	SYMBOL	CARD CODE
00	@	7-8	40	)	12-4-8
01	[	12-5-8	41	–	11
02	]	11-5-8	42	+	12
03	#	12-7-8	43	<	12-6-8
04	Δ	11-7-8	44	=	3-8
05	b	blank	45	>	6-8
06	A	12-1	46	&	2-8
07	B	12-2	47	$	11-3-8
10	C	12-3	50	*	11-4-8
11	D	12-4	51	(	0-4-8
12	E	12-5	52	%	0-5-8
13	F	12-6	53	:	5-8
14	G	12-7	54	?	12-0
15	H	12-8	55	!	11-0
16	I	12-9	56	,	0-3-8
17	J	11-1	57	\	0-6-8
20	K	11-2	60	0	0
21	L	11-3	61	1	1
22	M	11-4	62	2	2
23	N	11-5	63	3	3
24	O	11-6	64	4	4
25	P	11-7	65	5	5
26	Q	11-8	66	6	6
27	R	11-9	67	7	7
30	S	0-2	70	8	8
31	T	0-3	71	9	9
32	U	0-4	72	' apos.	4-8
33	V	0-5	73	;	11-6-8
34	W	0-6	74	/	0-1
35	X	0-7	75	.	12-3-8
36	Y	0-8	76	¤	0-7-8
37	Z	0-9	77	≠	0-2-8

ASCII TO FIELDATA CONVERSION

ASCII		FIELDATA		ASCII		FIELDATA	
OCTAL CODE	SYMBOL	OCTAL CODE	SYMBOL	OCTAL CODE	SYMBOL	OCTAL CODE	SYMBOL
40	SP	05	SP	106	F	13	F
41	!	55	!	107	G	14	G
42	"	76	¤	110	H	15	H
43	#	03	#	111	I	16	I
44	$	47	$	112	J	17	J
45	%	52	%	113	K	20	K
46	&	46	&	114	L	21	L
47	'	72	'	115	M	22	M
50	(	51	(	116	N	23	N
51	)	40	)	117	O	24	O
52	*	50	*	120	P	25	P
53	+	42	+	121	Q	26	Q
54	,	56	,	122	R	27	R
55	–	41	–	123	S	30	S
56	.	75	.	124	T	31	T
57	/	74	/	125	U	32	U
60	0	60	0	126	V	33	V
61	1	61	1	127	W	34	W
62	2	62	2	130	X	35	X
63	3	63	3	131	Y	36	Y
64	4	64	4	132	Z	37	Z
65	5	65	5	133	[	01	[
66	6	66	6	134	\	57	\
67	7	67	7	135	]	02	]
70	8	70	8	136	^	04	Δ
71	9	71	9	137	_	77	≠
72	:	53	:	140	`	00	@
73	;	73	;	141	a*	06	A**
74	<	43	<	through	through	through	through
75	=	44	=	172	z	37	Z
76	>	45	>	173	{	54	?
77	?	54	?	174	¦	57	\
100	@	00	@	175	}	55	!
101	A	06	A	176	~	04	Δ
102	B	07	B	177	DEL	77	≠
103	C	10	C				
104	D	11	D				
105	E	12	E				

*lowercase alphabet
**uppercase alphabet

THE SPECIAL CHARACTERS IN ASCII

SP designates space, which is normally nonprinting.
DEL designates delete, and has a code of all 1 bits.
This code obliterates any unwanted previous character – even on paper tape or other nonerasable medium.

The names of the 8 new special characters in ASCII are:	Some additional standardized names of interest:
" Quotation marks	# Number sign
^ Circumflex	& Ampersand
_ Underline	' Apostrophe
` Grave accent	* Asterisk
{ Opening brace	> Greater than sign
| Vertical line	@ At sign
} Closing brace	[Opening bracket
~ Tilde	\ Reverse slant

Definitions of the 32 ASCII control characters, codes 00_8 to 37_8:

00	NUL	Null – all zeros character which may serve as time fill
01	SOH	Start of heading
02	STX	Start of text
03	ETX	End of text
04	EOT	End of transmission
05	ENQ	Enquiry – "Who Are You?"
06	ACK	Acknowledge – "Yes"
07	BEL	Bell – human attention required
10	BS	Backspace
11	HT	Horizontal tabulation
12	LF	Line feed
13	VT	Vertical tabulation
14	FF	Form feed
15	CR	Carriage return
16	SO	Shift out – nonstandard code follows
17	SI	Shift in – return to standard code
20	DLE	Data link escape – change limited data communication controls
21	DC1	
22	DC2	
23	DC3	
24	DC4	
25	NAK	Negative acknowledge – "No"
26	SYN	Synchronous idle – from which to achieve synchronism
27	ETB	End of transmission block – relates to physical communication blocks
30	CAN	Cancel previous data
31	EM	End of medium – end of used, or wanted portion of information
32	SUB	Substitute character for one in error
33	ESC	Escape – for code extension – change some character interpretations
34	FS	File separator
35	GS	Group separator
36	RS	Record separator
37	US	Unit separator

format effectors for printing or punching

Device controls for turning on or off ancillary devices

These information separators are ordered in descending hierarchy. They are followed by ASCII 40_8 (space), which can also be thought of as a word separator.

INSTRUCTION REPERTOIRE

	Function Code (Octal) f j a	Mnemonic	Instruction	Description①
	00 0-17 –	–	Illegal Code	Causes illegal instruction interrupt to address 241_8
STORE	01 0-15 –	S,SA	Store A	(A) → U
	02 0-15 –	SN,SNA	Store Negative A	−(A) → U
	03 0-15 –	SM,SMA	Store Magnitude A	|(A)| → U
	04 0-15 –	S,SR	Store R	(R_a) → U
	05 0-15 –	SZ	Store Zero	ZEROS → U
	06 0-15 –	S,SX	Store X	(X_a) → U
	*07 00 –	SIA	Store Input Access Control Word	(A) → IACR; channel number per U_5–0
	*07 01 –	SOA	Store Output Access Control Word	(A) → OACR; channel number per U_5–0
	*07 02 –	SIP	Store Input Pointer Word	(A) → ICPR; channel number per U_5–0
	*07 03 –	SOP	Store Output Pointer Word	(A) → OCPR; channel number per U_5–0
	*07 04 –	LIA	Load Input Access Control Word	(IACR) → A; channel number per U_5–0
	*07 05 –	LOA	Load Output Access Control Word	(OACR) → A; channel number per U_5–0
	*07 06 –	LIP	Load Input Pointer Word	(ICPR) → A; channel number U_5–0
	*07 07 –	LOP	Load Output Pointer Word	(OCPR) → A; channel number per U_5–0
	*07 10 0	LCB	Load Chain Base Register IOAU 0	(U) 14–0 → CBR of IOAU 0 for channel 0–23
	*07 10 1	LCB	Load Chain Base Register IOAU 1	(U) 14–0 → CBR of IOAU 1 for channels 24–47
	*07 11 0	LPI	Load Processor Interrupt Pointer Register IOAU 0	(U) 1–0 → PIP register of IOAU 0 for channels 0–23
	*07 11 1	LPI	Load Processor Interrupt Pointer Register IOAU 1	(U) 1–0 → PIP register of IOAU 1 for channels 24–47
LOAD BANK AND JUMP	†07 12 –	LDJ	Load D-Bank Base and Jump	BDW → active PSR,SLR (D-bank value); BDI register D-bank value → Xa_{35-18}; new BDI values → BDI register; relative P+1 → Xa_{17-0}; jump to U
	†07 13 –	LIJ	Load I-Bank Base and Jump	BDW → active PSR, SLR (I-bank values); BDI register I-bank values → Xa_{35-18}; new BDI values → BDI register; relative P+1 → Xa_{17-0}; jump to U

*1110 only
†1110 instruction simulated on 1106/1108
▨ = Restricted use (illegal code or privileged use by the executive)

	Function Code (Octal) f j a	Mnemonic	Instruction	Description ①
PSR DESIGNATORS	†07 14 –	LPD	Load PSR Designators	$U_{6,5,3-0} \rightarrow$ PSRM; Bit 6 → D20 Bit 2 → D8 Bit 5 → D17 Bit 1 → D5 Bit 3 → D10 Bit 0 → D4
	†07 15 –	SPD	Store PSR Designators	PSRM D-bits → U_{6-0}; D20 → Bit 6 D8 → Bit 2 D17 → Bit 5 D5 → Bit 1 D12 → Bit 4 D4 → Bit 0 D10 → Bit 3
	*07 16 0	LBR	Load Breakpoint Register	(U) → Breakpoint Register
	*07 16 1	SJS	Store Jump History Stack	(Jump History Stack) → U, repeat
LOAD A	10 0-17 –	L,LA	Load A	(U) → A
	11 0-17 –	LN,LNA	Load Negative A	−(U) → A
	12⑤	LM,LMA	Load Magnitude A	|(U)| → A
	13⑤	LNM,LNMA	Load Negative Magnitude A	−|(U)| → A
ADD A	14 0-17 –	A,AA	Add To A	(A)+(U) → A
	15 0-17 –	AN,ANA	Add Negative to A	(A)−(U) → A
	16⑤	AM,AMA	Add Magnitude To A	(A)+|(U)| → A
	17⑤	ANM,ANMA	Add Negative Magnitude to A	(A)−|(U)| → A
	20 0-17 –	AU	Add Upper	(A)+(U) → A+1
	21 0-17 –	ANU	Add Negative Upper	(A)−(U) → A+1
BT	22 0-15 –	BT	Block Transfer	(X_x+u) → X_a+u; repeat(R1) times
LR	23 0-17 –	L,LR	Load R	(U) → R_a
ADD X	24 0-17 –	A,AX	Add To X	(X_a)+(U) → X_a
	25 0-17 –	AN,ANX	Add Negative To X	(X_a)−(U) → X_a
LOAD X	26 0-17 –	LXM	Load X Modifier	(U) → $X_{a_{17-0}}$; $X_{a_{35-18}}$ unchanged
	27 0-17 –	L,LX	Load X	(U) → X_a
MULTIPLY	30 0-17 –	MI	Multiply Integer	(A)*(U) → A,A+1
	31 0-17 –	MSI	Multiply Single Integer	(A)*(U) → A
	32 0-17 –	MF	Multiply Fractional	(A)*(U) → A,A+1, left shift circular 1 bit
BYTE MANIPULATION	*33 00 –	BM	Byte move	(SJ0) → SJ1
	*33 01 –	BMT	Byte Move and Translate	Translated (SJ0) → SJ1
	*33 02 –	BTT	Byte Translate and Test	Translate (SJ0); test against (A); terminate if ≠
	*33 03 –	BTC	Byte Translate and Compare	Translated (SJ0) − translated (SJ1) → A; terminate if (A) ≠ 0
	*33 04 –	BC	Byte Compare	(SJ0) − (SJ1) → A; terminate if (A) ≠ 0

†1110 instruction simulated on 1106/1108
*1110 only
▨ = Restricted use (illegal code or privileged use by the executive)

	Function Code (Octal) f j a	Mnemonic	Instruction	Description[1]
BYTE MANIPULATION	*33 05 –	BPD	Byte to Packed Decimal Convert	Convert (SJ0) → packed decimal SJ1
	*33 06 –	PDB	Packed Decimal to Byte Convert	Convert packed decimal (SJ0) → SJ1
	*33 07 –	EDIT	Edit	Edit (SJ0) under control of (SJ2) → SJ1
	*33 10 –	BI	Byte to Binary Single Integer Convert	Convert (SJ0) → signed binary integer A
	*33 11 –	BDI	Byte to Binary Double Integer Convert	Convert (SJ0) → signed binary integer A,A+1
	*33 12 –	IB	Binary Single Integer to Byte Convert	Convert signed binary integer (A) → SJ0
	*33 13 –	DIB	Binary Double Integer to Byte Convert	Convert signed binary integer (A,A+1) → SJ0
	*33 14 –	BF	Byte to Single Floating-Point Convert	Convert (SJ0) → single-precision floating-point format A
	*33 15 –	BDF	Byte to Double Floating-Point Convert	Convert (SJ0) → double precision floating-point format A,A+1
	*33 16 –	FB	Single Floating-Point to Byte Convert	Convert single-precision floating-point format (A) → SJ0
	*33 17 –	DFB	Double Floating-Point to Byte Convert	Convert double-precision floating-point format (A,A+1) → SJ0
DIVIDE	34 0-17 –	DI	Divide Integer	(A,A+1) ÷ (U) → A; REMAINDER A+1
	35 0-17 –	DSF	Divide Single Fractional	[(A,36 sign bits) right algebraic shift 1 place] ÷ (U) → A+1
	36 0-17 –	DF	Divide Fractional	[(A,A+1) right algebraic shift 1 place] ÷ (U) → A; REMAINDER → A+1
BYTE MANIPULATION	*37 00 –	QB	Quarter-Word Byte to Binary Compress	Discard $(A)_{35}$, $(A)_{26}$, $(A)_{17}$, and $(A)_{8}$; remaining bits (A) → A_{31-0}; $(A)_{31}$ → A_{35-32}
	*37 01 –	BQ	Binary to Quarter-Word Byte Extend	Discard $(A)_{35-32}$; $(A)_{31-0}$ → A_{34-27}, A_{25-18}, A_{16-9}, and A_{7-0}; zero fill A_{35}, A_{26}, A_{17}, and A_{8}
	*37 02 –	QBH	Quarter Word Byte to Binary Halves Compress	Discard $(A)_{35}$, $(A)_{26}$, $(A)_{17}$, and $(A)_{8}$; remaining bits (A) → A_{33-18} and $_{15-0}$; $(A)_{33}$ → A_{35-34}, $(A)_{15}$ → A_{17-16}
	*37 03 –	BHQ	Binary Halves to Quarter-Word Byte Extend	Discard $(A)_{35-34}$ and $(A)_{17-16}$; remaining bits (A) → A_{34-27}, A_{25-18}, A_{16-9}, and A_{7-0}; zero-fill A_{35}, A_{26}, A_{17}, and A_{8}

*1110 only

	Function Code (Octal) f j a	Mnemonic	Instruction	Description①
BYTE MANIPULATION	*37 04 –	QDB	Quarter-Word Byte to Double Binary Compress	Discard A_{35}, A_{26}, A_{17}, A_8, $A{+}1_{35}$, $A{+}1_{26}$, $A{+}1_{17}$, and $A{+}1_8$; remaining bits $(A,A{+}1) \rightarrow A_{27-0}$ and $A{+}1$; $(A)_{27} \rightarrow A_{35-28}$
	*37 05 –	DBQ	Double Binary to Quarter-Word Byte Extend	Discard $(A)_{35-28}$; remaining bits $(A,A{+}1) \rightarrow A_{34-27}$, A_{25-18}, A_{16-9}, A_{7-0}, $A{+}1_{34-27}$, $A{+}1_{25-18}$, $A{+}1_{16-9}$, and $A{+}1_{7-0}$; zero-fill A_{35}, A_{26}, A_{17}, A_8, $A{+}1_{35}$, $A{+}1_{26}$, $A{+}1_{17}$, and $A{+}1_8$
	*37 06 –	BA	Byte Add	(SJ1) + (SJ0) →SJ2
	*37 07 –	BAN	Byte Add Negative	(SJ1) – (SJ0) →SJ2
	37 10-17 –		Illegal Code	Causes illegal instruction interrupt to address 241_8
LOGICAL	40 0-17 –	OR	Logical OR	(A) OR (U)→A+1
	41 0-17 –	XOR	Logical Exclusive OR	(A) XOR (U)→A+1
	42 0-17 –	AND	Logical AND	(A) AND (U)→A+1
	43 0-17 –	MLU	Masked Load Upper	[(U) AND (R2)] OR [(A) AND (R2)']→A+1
TEST	44 0-17 –	TEP	Test Even Parity	Skip NI if (U) AND (A) has even parity④
	45 0-17 –	TOP	Test Odd Parity	Skip NI if (U) AND (A) has odd parity③
LXI	46 0-17 –	LXI	Load X Increment	$(U) \rightarrow X_{a_{35-18}}$; $X_{a_{17-0}}$ unchanged
TEST	47 0-17 –	TLEM	Test Less Than or Equal To Modifier	Skip NI if $(U)_{17-0} \leqslant (X_a)_{17-0}$; always $(X_a)_{17-0} + (X_a)_{35-18} \rightarrow X_{a_{17-0}}$
		TNGM	Test Not Greater Than Modifier	
	50 0-17 –	TZ	Test Zero	Skip NI if (U) = ±0
	51 0-17 –	TNZ	Test Nonzero	Skip NI if (U) ≠ ±0
	52 0-17 –	TE	Test Equal	Skip NI if (U) = (A)
	53 0-17 –	TNE	Test Not Equal	Skip NI if (U) ≠ (A)
	54 0-17 –	TLE	Test Less Than or Equal	Skip NI if (U) ⩽ (A)
		TNG	Test Not Greater	
	55 0-17 –	TG	Test Greater	Skip NI if (U) > (A)
	56 0-17 –	TW	Test Within Range	Skip NI if (A) < (U) ⩽ (A+1)
	57 0-17 –	TNW	Test Not Within Range	Skip NI if (U) ⩽ (A) or (U) > (A+1)
	60 ⑤	TP	Test Positive (requires full word or sign extension)	Skip NI if $(U)_{35} = 0$
	61 ⑤	TN	Test Negative (requires full word or sign extension)	Skip NI if $(U)_{35} = 1$

*1110 only
▒▒ = Restricted use (illegal code or privileged use by the executive)

	Function Code (Octal) f j a	Mnemonic	Instruction	Description①
SEARCH	62 0-17 –	SE	Search Equal	Skip NI if (U) = (A), else repeat
	63 0-17 –	SNE	Search Not Equal	Skip NI if (U) ≠ (A), else repeat
	64 0-17 –	SLE SNG	Search Less Than or Equal Search Not Greater	Skip NI if (U) ≤ (A), else repeat
	65 0-17 –	SG	Search Greater	Skip NI if (U) > (A), else repeat
	66 0-17 –	SW	Search Within Range	Skip NI if (A) < (U) ≤ (A+1), else repeat
	67 0-17 –	SNW	Search Not Within Range	Skip NI if (U) ≤ (A) or (U) > (A+1), else repeat
JGD	70 0-177 ②	JGD	Jump Greater and Decrement	Jump to U if (Control Register)$_{ja}$ > 0; go to NI if (Control Register)$_{ja}$ ≤ 0; always (Control Register)$_{ja}$ −1 → Control Register $_{ja}$
MASKED SEARCH	71 00 –	MSE	Mask Search Equal	Skip NI if (U) AND (R2)=(A) AND (R2), else repeat
	71 01 –	MSNE	Mask Search Not Equal	Skip NI if (U) AND (R2) ≠ (A) AND (R2), else repeat
	71 02 –	MSLE MSNG	Mask Search Less Than or Equal Mask Search Not Greater	Skip NI if (U) AND (R2) ≤ (A) AND (R2), else repeat
	71 03 –	MSG	Mask Search Greater	Skip NI if (U) AND (R2) > (A) AND (R2), else repeat
	71 04 –	MSW	Masked Search Within Range	Skip NI if (A) AND (R2) < (U) AND (R2) ≤ (A+1) AND (R2), else repeat
	71 05 –	MSNW	Masked Search Not Within Range	Skip NI if (U) AND (R2) ≤ (A) AND (R2) or (U) AND (R2) > (A+1) AND (R2), else repeat
	71 06 –	MASL	Masked Alpha-numeric Search Less Than or Equal	Skip NI if (U) AND (R2) ≤ (A) AND (R2), else repeat
	71 07 –	MASG	Masked Alpha-numeric Search Greater	Skip NI if (U) AND (R2) > (A) AND (R2), else repeat
DOUBLE PRECISION	71 10 –	DA	Double Precision Fixed-Point Add	(A,A+1)+(U,U+1) → A,A+1
	71 11 –	DAN	Double Precision Fixed-Point Add Negative	(A,A+1)−(U,U+1) → A,A+1
	71 12 –	DS	Double Store A	(A,A+1) → U,U+1
	71 13 –	DL	Double Load A	(U,U+1) → A,A+1
	71 14 –	DLN	Double Load Negative A	−(U,U+1) → A,A+1
	71 15 –	DLM	Double Load Magnitude A	|(U,U+1)| → A,A+1
	71 16 –	DJZ	Double Precision Jump Zero	Jump to U if (A,A+1)= ±0; go to NI if (A,A+1) ≠ ±0
	71 17 –	DTE	Double Precision Test Equal	Skip NI if (U,U+1)=(A,A+1)

	Function Code (Octal) f j a	Mnemonic	Instruction	Description①
	72 00 –	–	Illegal Code	Causes illegal instruction interrupt to address 241_8
JUMP	72 01 –	SLJ	Store Location and Jump	(P)–BASE ADDRESS MODIFIER [BI or BD] $\rightarrow U_{17-0}$; jump to U+1
JUMP	72 02 –	JPS	Jump Positive and Shift	Jump to U if $(A)_{35}=0$; go to NI if $(A)_{35}=1$; always shift (A) left circularly one bit position.
JUMP	72 03 –	JNS	Jump Negative and Shift	Jump to U if $(A)_{35}=1$; go to NI if $(A)_{35}=0$; always shift (A) left circularly one bit position
PARTIAL WORD ADD	72 04 –	AH	Add Halves	$(A)_{35-18}+(U)_{35-18} \rightarrow A_{35-18}$; $(A)_{17-0}+(U)_{17-0} \rightarrow A_{17-0}$
PARTIAL WORD ADD	72 05 –	ANH	Add Negative Halves	$(A)_{35-18}-(U)_{35-18} \rightarrow A_{35-18}$; $(A)_{17-0}-(U)_{17-0} \rightarrow A_{17-0}$
PARTIAL WORD ADD	72 06 –	AT	Add Thirds	$(A)_{35-24}+(U)_{35-24} \rightarrow A_{35-24}$; $(A)_{23-12}+(U)_{23-12} \rightarrow A_{23-12}$; $(A)_{11-0}+(U)_{11-0} \rightarrow A_{11-0}$
PARTIAL WORD ADD	72 07 –	ANT	Add Negative Thirds	$(A)_{35-24}-(U)_{35-24} \rightarrow A_{35-24}$; $(A)_{23-12}-(U)_{23-12} \rightarrow A_{23-12}$; $(A)_{11-0}-(U)_{11-0} \rightarrow A_{11-0}$
EX	72 10 –	EX	Execute	Execute the instruction at U
ER	72 11 –	ER	Executive Request	Causes executive request interrupt to address 242_8
	72 12 –	–	Illegal Code	Causes illegal instruction interrupt to address 241_8
	72 13 –	PAIJ	Prevent All I/O Interrupts and Jump	Prevent all I/O interrupts and jump to U
SCN	72 14 0-3	SCN	Store Channel Number	If a=0: CHANNEL NUMBER $\rightarrow U_{3-0}$; If a=1: CHANNEL NUMBER $\rightarrow U_{3-0}$ and CPU NUMBER $\rightarrow U_{5-4}$ If a=2: CHANNEL NUMBER $\rightarrow U_{5-0}$; If a=3: CHANNEL NUMBER $\rightarrow U_{5-0}$ and CAU NUMBER $\rightarrow U_{14-12}$
	72 15 0	LPS	Load Processor State Register	(U) →PSR (U) →PSRM0
	1	LMP*	Load Main Processor State Register	(U,U+1) →PSRM0, PSRM1
	2	LUP*	Load Utility Processor State Register	(U,U+1) →PSRU0, PSRU1
	72 16 0	LSL	Load Main Storage Limits Register	(U) →SLRM
	1	LUS*	Load Utility Storage Limits Register	(U) →SLRU
	2	SL*	Store Main Storage Limits Register	(SLRM) →U
	3	SUL*	Storage Utility Storage Limits Register	(SLRU) →U

*1110 only
▨ = Restricted use (illegal code or privileged use by the executive)

	Function Code (Octal) f	j	a	Mnemonic	Instruction	Description ①
	72	17	–	–	Illegal Code	Causes illegal instruction interrupt to address 241_8
SHIFT	73	00	–	SSC	Single Shift Circular	Shift (A) right circularly U places
	73	01	–	DSC	Double Shift Circular	Shift (A,A+1) right circularly U places
	73	02	–	SSL	Single Shift Logical	Shift (A) right U places; zero-fill
	73	03	–	DSL	Double Shift Logical	Shift (A,A+1) right U places; zero-fill
	73	04	–	SSA	Single Shift Algebraic	Shift (A) right U places; sign-fill
	73	05	–	DSA	Double Shift Algebraic	Shift (A,A+1) right U places; sign-fill
	73	06	–	LSC	Load Shift and Count	(U) → A, shift (A) left circularly until $(A)_{35} \neq (A)_{34}$ number of shifts → A+1
	73	07	–	DSLC	Double Load Shift and Count	(U,U+1) → A,A+1; shift (A,A+1) left circularly until $(A,A+1)_{71} \neq (A,A+1)_{70}$; number of shifts → A+2
	73	10	–	LSSC	Left Single Shift Circular	Shift (A) left circularly U places
	73	11	–	LDSC	Left Double Shift Circular	Shift (A,A+1) left circularly U places
	73	12	–	LSSL	Left Single Shift Logical	Shift (A) left U places zero-fill
	73	13	–	LDSL	Left Double Shift Logical	Shift (A,A+1) left U places; zero-fill
	73	14	0-5	III	Initiate Interprocessor Interrupt	Initiate interprocessor interrupt per a; a=0: interrupt CAU number 0; a=1: interrupt CAU number 1; a=2: interrupt CAU number 2; a=3: interrupt CAU number 3; a=4: interrupt CAU number 4; a=5: interrupt CAU number 5
	73	14	10	ESDC (1110)	Enable Second Day Clock	Enable day clock in IOAU having channels 24–47
	73	14	10	ALRM (1106/1108)	Alarm	Turn on alarm
	73	14	11	EDC	Enable Day Clock	Enable day clock (1106/1108) Enable day clock in IOAU having channels 0–23 (1110)
	73	14	12	DDC	Disable Day Clock	Disable day clock
	73	14	13	CES*	Clear and Enable Storage Reference Counters	Clear storage reference counters to zero and enable on NI
	73	14	14	ES*	Enable Storage Reference Counters	Enable storage reference counters on NI

*1110 only.
▭ = Restricted use (illegal code or privileged use by the executive)

	Function Code (Octal) f	j	a	Mnemonic	Instruction	Description ①
	73	15	–	SIL	Select Interrupt Locations	(a) → MSR (1106/1108) $(U)_{8-0}$ → MSR (1110)
	73	16	0	LCR	Load Channel Select Register	$(U)_{3-0}$ → CSR (1106/1108) $(U)_{5-0}$ → CSR, if $(U)_9$=1, select back-to-back data transfer mode (1110)
	73	16	1	LLA	Load Last Address Register	$(U)_{2-0}$ → LAR (1106/1108) $(U)_{8-0}$ → LAR (1110)
TEST AND SET	73	17	0	TS	Test and Set	If $(U)_{30}$=1, interrupt to MSR+244_8; if $(U)_{30}$=0, go to NI; always 01_8 → U_{35-0}
	73	17	1	TSS*	Test and Set and Skip	If $(U)_{30}$=0, skip NI; if $(U)_{30}$=1, go to NI; always 01_8 → U_{35-30}
	73	17	2	TCS*	Test and Clear and Skip	If $(U)_{30}$=0, go to NI; if $(U)_{30}$=1, skip NI; always clear $(U)_{35-30}$
JUMP	74	00	–	JZ	Jump Zero	Jump to U if (A)=±0; go to NI if (A)≠±0
	74	01	–	JNZ	Jump Nonzero	Jump to U if (A)≠±0; go to NI if (A)=±0
	74	02	–	JP	Jump Positive	Jump to U if $(A)_{35}$=0; go to NI if $(A)_{35}$=1
	74	03	–	JN	Jump Negative	Jump to U if $(A)_{35}$=1; go to NI if $(A)_{35}$=0
	74	04	–	JK J	Jump Keys Jump	Jump to U if a=0 or if a≠ set SELECT JUMPS control circuit; go to NI if neither is true
	74	05	–	HKJ HJ	Halt Keys and Jump Halt Jump	Stop if a=0 or if [a AND set SELECT JUMPS control circuit] ≠0; on restart or continuation, jump to U
NOP	74	06	–	NOP	No Operation	Proceed to next instruction; if the x and h fields of the instruction operand are non-zero, index register incrementation occurs
JUMP	74	07	–	AAIJ	Allow All I/O Interrupts and Jump	All all I/O interrupts and jump to U
	74	10	–	JNB	Jump No Low Bit	Jump to U if $(A)_0$=0; go to NI if $(A)_0$=1
	74	11	–	JB	Jump Low Bit	Jump to U if $(A)_0$=1; go to NI if $(A)_0$=0
	74	12	–	JMGI	Jump Modifier Greater and Increment	Jump to U if $(X_a)_{17-0}$>0; go to NI if $(X_a)_{17-0}$≤0; always $(X_a)_{17-0}$ + $(X_a)_{35-18}$ → $X_{a_{17-0}}$
	74	13	–	LMJ	Load Modifier and Jump	(P)–BASE ADDRESS MODIFIER [BI or BD] → $X_{a_{17-0}}$; jump to U
	74	14	0	JO	Jump Overflow	Jump to U if D1 of PSR=1; go to NI if D1=0
	74	14	1	JFU*	Jump Floating Underflow	Jump to U if D21=1, clear D21; go to NI if D21=0
	74	14	2	JFO*	Jump Floating Overflow	Jump to U if D22=1, clear D22; go to NI if D22=0

*1110 only.
▨ = Restricted use (illegal code or privileged use by the executive)

	Function Code (Octal) f	j	a	Mnemonic	Instruction	Description ①
JUMP	74	14	3	JDF*	Jump Divide Fault	Jump to U if D23=1, clear D23; go to NI if D23=0
	74	15	0	JNO	Jump No Overflow	Jump to U if D1=0; go to NI if D1=1
	74	15	1	JNFU*	Jump No Floating Underflow	Jump to U if D21=0; go to NI if D21=1; clear D21
	74	15	2	JNFO*	Jump No Floating Overflow	Jump to U if D22=0; go to NI if D22=1; clear D22
	74	15	3	JNDF*	Jump No Divide Fault	Jump to U if D23=0; go to NI if D23=1; clear D23
	74	16	–	JC	Jump Carry	Jump to U if D0 of PSR=1; go to NI if D0=0
	74	17	–	JNC	Jump No Carry	Jump to U if D0 of PSR=0; go to NI if D0=1
	75	00	–	LIC	Load Input Channel	For channel [a [redacted] CSR]: (U) →IACR; set input active; clear input monitor
	75	01	–	LICM	Load Input Channel and Monitor	For channel [a [redacted] CSR]: (U) →IACR; set input active; set input monitor
	75	02	–	JIC	Jump On Input Channel Busy	Jump to U if input active is set for channel [a [redacted] CSR]; go to NI if input active is clear
	75	03	–	DIC	Disconnect Input Channel	For channel [a [redacted] CSR]: clear input active; clear input monitor
	75	04	–	LOC	Load Output Channel	For channel [a [redacted] CSR]: (U) →OACR; set output active; clear output monitor; clear function active (ISI only)
	75	05	–	LOCM	Load Output Channel and Monitor	For channel [a [redacted] CSR]: (U) →OACR; set output active; set output monitor; clear function active (ISI only)
	75	06	–	JOC	Jump On Output Channel Busy	Jump to U if output active is set for channel [a [redacted] CSR]; go to NI if output active is clear
	75	07	–	DOC	Disconnect Output Channel	For channel [a [redacted] CSR]: clear output active; clear output monitor; clear external function, or 1110, clear function active (ISI only)
	75	10	–	LFC	Load Function In Channel	For channel [a [redacted] CSR]: (U) →OACR; set output active (ISI only), external function, or 1110, function active (ISI only) and force external function; clear output monitor (ISI only)

*1110 only.
[redacted] = Restricted use (illegal code or privileged use by the executive)

	Function Code (Octal) f	j	a	Mnemonic	Instruction	Description ①
	75	11	–	LFCM	Load Function In Channel and Monitor	For channel [a CSR]: (U) → OACR; set output active (ISI only), external function, or 1110, function active (ISI only), force external function, and output monitor (ISI only)
	75	12	–	JFC	Jump On Function In Channel	Jump to U if force external function is set for channel [a CSR]; go to NI if force external function is clear
	75	13	–	–	Illegal Code ⑥	
	75	14	–	AACI	Allow All Channel External Interrupts	Allow all external interrupts
	75	15	–	PACI	Prevent All Channel External Interrupts	Prevent all external interrupts
	75	16	0	ACI*	Allow Channel Interrupts	Allow interrupts on channels 23–0 specified by one bits in $(U)_{23-0}$
	75	16	1	ACI*	Allow Channel Interrupts	Allow interrupts on channels 47–24 specified by one bits in $(U)_{23-0}$
	75	17	0	PCI*	Prevent Channel Interrupts	Prevent interrupts on channels 23–0 specified by one bits in $(U)_{23-0}$
	75	17	1	PCI*	Prevent Channel Interrupts	Prevent interrupts on channels 47–24 specified by one bits in $(U)_{23-0}$
FLOATING POINT	76	00	–	FA	Floating Add	(A)+(U) → A; RESIDUE → A+1 (if D17=1*)
	76	01	–	FAN	Floating Add Negative	(A)–(U) → A; RESIDUE → A+1 (if D17=1*)
	76	02	–	FM	Floating Multiply	(A)·(U) → A,A+1 (if D17=1*)
	76	03	–	FD	Floating Divide	(A) ÷ (U) → A; REMAINDER → A+1 (if D17=1*)
	76	04	–	LUF	Load and Unpack Floating	$\|(U)\|_{34-27} \rightarrow A_{7-0}$, zero-fill; $(U)_{26-0} \rightarrow A{+}1_{26-0}$; sign-fill
	76	05	–	LCF	Load and Convert To Floating	$(U)_{35} \rightarrow A{+}1_{35}$; [NORMALIZED $(U)]_{26-0} \rightarrow A{+}1_{26-0}$; if $(U)_{35}$=0, $(A)_{7-0}$ ± NORMALIZING COUNT → $A{+}1_{34-27}$; if $(U)_{35}$=1, ones complement of [$(A)_{7-0}$ ± NORMALIZING COUNT] → $A{+}1_{34-27}$
	76	06	–	MCDU	Magnitude of Characteristic Difference To Upper	$\|\|(A)\|_{35-27} - \|(U)\|_{35-27}\| \rightarrow A{+}1_{8-0}$; ZEROS → $A{+}1_{35-9}$
	76	07	–	CDU	Characteristic Difference To Upper	$\|(A)\|_{35-27} - \|(U)\|_{35-27} \rightarrow A{+}1_{8-0}$; SIGN BITS → $A{+}1_{35-9}$

*1110 only
▨ = Restricted use (illegal code or privileged use by the executive)

	Function Code (Octal) f	j	a	Mnemonic	Instruction	Description①
FLOATING POINT	76	10	–	DFA	Double Precision Floating Add	(A,A+1)+(U,U+1) →A,A+1
	76	11	–	DFAN	Double Precision Floating Add Negative	(A,A+1)–(U,U+1) →A,A+1
	76	12	–	DFM	Double Precision Floating Multiply	(A,A+1)·(U,U+1) → A,A+1
	76	13	–	DFD	Double Precision Floating Divide	(A,A+1)÷(U,U+1) →A,A+1
	76	14	–	DFU	Double Load and Unpack Floating	$\|(U)\|_{34-24}$ →A_{10-0}, zero-fill; $(U)_{23-0}$ → $A+1_{23-0}$, sign-fill; (U+1) →A+2
	76	15	–	DLCF,DFP	Double Load and Convert To Floating	$(U)_{35}$ →$A+1_{35}$; [NORMALIZED (U,U+1)]$_{59-0}$ →$A+1_{23-0}$ and A+2; if $(U)_{35}$=0,$(A)_{10-0}$ ± NORMALIZING COUNT →$A+1_{34-24}$; if $(U)_{35}$=1, ones complement of [$(A)_{10-0}$ ± NORMALIZING COUNT] →$A+1_{34-24}$
	76	16	–	FEL	Floating Expand and Load	If $(U)_{35}$=0, $(U)_{35-27}$+ 1600_8 →A_{35-24}; if $(U)_{35}$=1, $(U)_{35-27}$− 1600_8 →A_{35-24}; $(U)_{26-3}$ →A_{23-0}; $(U)_{2-0}$ →$A+1_{35-33}$; $(U)_{35}$ →$A+1_{32-0}$
	76	17	–	FCL	Floating Compress and Load	If $(U)_{35}$=0, $(U)_{35-24}$− 1600_8 →A_{35-27}; if $(U)_{35}$=1, $(U)_{35-24}$+ 1600_8 →A_{35-27}; $(U)_{23-0}$ →A_{26-3}; $(U+1)_{35-33}$ →A_{2-0}
	77	0-17		–	Illegal Code	Causes illegal instruction interrupt to address 241_8

▨ = Restricted use (illegal code or privileged use by the executive)

NOTES:

① NI stands for Next Instruction.

② The a and j fields together serve to specify any of the 128 control registers, meaning any register from the full X, A or R set.

③ If only one specific bit of (U) is set to 1, TOP skips if the corresponding bit of (A) is also set to 1, regardless of what other bits of (A) are set.

④ If only one specific bit of (U) is set to 1, TEP will skip if the corresponding bit of (A) is zero, regardless of what other bits of (A) are set.

⑤ j=0_8, 3_8–7_8, 17_8 are the only meaningful j designator values because of the sign extension requirements. j=4_8–7_8 is not meaningful for quarter-word mode.

⑥ If guard mode is set, causes guard mode interrupt to address 243_8. If guard mode is not set, same as NOP.

MNEMONIC/FUNCTION CODE CROSS-REFERENCE

Mnemonic	Function Code (Octal) f	j	a	Mnemonic	Function Code (Octal) f	j	a
A	14			ES	73	14	14
A	24			ESDC	73	14	10
AA	14			EX	72	10	
AACI	75	14					
AAIJ	74	07		FA	76	00	
ACI	75	16		FAN	76	01	
AH	72	04		FB	33	16	
ALRM	73	14	10	FCL	76	17	
AM	16			FD	76	03	
AMA	16			FEL	76	16	
AN	15			FM	76	02	
AN	25						
ANA	15			HJ	74	05	
AND	42			HKJ	74	05	
ANH	72	05					
ANM	17			IB	33	12	
ANMA	17			III	73	14	0–5
ANT	72	07					
ANU	21			J	74	04	
ANX	25			JB	74	11	
AT	72	06		JC	74	16	
AU	20			JDF	74	14	3
AX	24			JFC	75	12	
				JFO	74	14	2
BA	37	06		JFU	74	14	1
BAN	37	07		JGD	70		
BC	33	04		JIC	75	02	
BDF	33	15		JK	74	04	
BDI	33	11		JMGI	74	12	
BF	33	14		JN	74	03	
BHQ	37	03		JNB	74	10	
BI	33	10		JNC	74	17	
BM	33	00		JNDF	74	15	3
BMT	33	01		JNFO	74	15	2
BPD	33	05		JNFU	74	15	1
BQ	37	01		JNO	74	15	0
BT	22	0-15		JNS	72	03	
BTC	33	03		JNZ	74	01	
BTT	33	02		JO	74	14	0
				JOC	75	06	
CDU	76	07		JP	74	02	
CES	73	14	13	JPS	72	02	
				JZ	74	00	
DA	71	10					
DAN	71	11		L	10		
DBQ	37	05		L	23		
DDC	73	14	12	L	27		
DF	36			LA	10		
DFA	76	10		LBR	07	16	0
DFAN	76	11		LCB	07	10	
DFB	33	17		LCF	76	05	
DFD	76	13		LCR	73	16	0
DFM	76	12		LDJ	07	12	
DFP	76	15		LDSC	73	11	
DFU	76	14		LDSL	73	13	
DI	34			LFC	75	10	
DIB	33	13		LFCM	75	11	
DIC	75	03		LIA	07	04	
DJZ	71	16		LIC	75	00	
DL	71	13		LICM	75	01	
DLCF	76	15		LIJ	07	13	
DLM	71	15		LIP	07	06	
DLN	71	14		LLA	73	16	1
DLSC	73	07		LM	12		
DOC	75	07		LMA	12		
DS	71	12		LMP	72	15	1
DSA	73	05		LMJ	74	13	
DSC	73	01		LN	11		
DSF	35			LNA	11		
DSL	73	03		LNM	13		
DTE	71	17		LNMA	13		
				LOA	07	05	
EDC	73	14	11	LOC	75	04	
EDIT	33	07		LOCM	75	05	
ER	72	11		LOP	07	07	

Mnemonic	Function Code (Octal)		
	f	j	a
LPD	07	14	
LPI	07	11	
LPS	72	15	0
LR	23		
LSC	73	06	
LSL	72	16	0
LSSC	73	10	
LSSL	73	12	
LUF	76	04	
LUP	72	15	2
LUS	72	16	1
LX	27		
LXI	46		
LXM	26		
MASG	71	07	
MASL	71	06	
MCDU	76	06	
MF	32		
MI	30		
MLU	43		
MSE	71	00	
MSG	71	03	
MSI	31		
MSLE	71	02	
MSNE	71	01	
MSNG	71	02	
MSNW	71	05	
MSW	71	04	
NOP	74	06	
OR	40		
PACI	75	15	
PAIJ	72	13	
PCI	75	17	
PDB	33	06	
QB	37	00	
QBH	37	02	
QDB	37	04	
S	01	0-15	
S	04	0-15	
S	06	0-15	
SA	01	0-15	
SCN	72	14	
SE	62		
SG	65		
SIA	07	00	
SIP	07	02	
SJS	07	16	1

Mnemonic	Function Code (Octal)		
	f	j	a
SL	72	16	2
SLE	64		
SLJ	72	01	
SM	03	0-15	
SMA	03	0-15	
SN	02	0-15	
SNA	02	0-15	
SNE	63		
SNG	64		
SNW	67		
SOA	07	01	
SOP	07	03	
SPD	07	15	
SR	04	0-15	
SSA	73	04	
SSC	73	00	
SSL	73	02	
SUL	72	16	3
SW	66		
SX	06	0-15	
SZ	05	0-15	
TCS	73	17	2
TE	52		
TEP	44		
TG	55		
TLE	54		
TLEM	47		
TN	61		
TNE	53		
TNG	54		
TNGM	47		
TNW	57		
TNZ	51		
TOP	45		
TP	60		
TS	73	17	0
TSS	73	17	1
TW	56		
TZ	50		
XOR	41		

OCTAL VS. MNEMONIC INSTRUCTION CODES

FIRST DIGIT	FUNCTION CODE – SECOND DIGIT							
	0	1	2	3	4	5	6	7
0		S,SA	SN SNA	SM SMA	S SR	SZ	S,SX	I/O (see below)
1	L LA	LN LNA	LM LMA	LNMA	A AA	AN ANA	AM AMA	ANM ANMA
2	AU	ANU	BT	L,LR	A AX	AN ANX	LXM	L LX
3	MI	MSI	MF	Bytes (see below)	DI	DSF	DF	Bytes (see below)
4	OR	XOR	AND	MLU	TEP	TOP	LXI	TLEM TNGM
5	TZ	TNZ	TE	TNE	TLE TNG	TG	TW	TNW
6	TP	TN	SE	SNE	SLE SNG	SG	SW	SNW
7	JGD	SEE BELOW						

FUNCT. CODE	FIRST j DIGIT	SECOND j DIGIT							
		0	1	2	3	4	5	6	7
07	0 1	SIA LCB	SOA LPI	SIP LDJ	SOP LIJ	LIA LPD	LOA SPD	LIP LBR, SJS	LOP
33	0 1	BM BI	BMT BDI	BTT* IB	BTC DIB	BC BF	BPD* BDF	PDB* FB	EDIT DFB
37	0* 1	QB*	BQ*	QBH*	BHQ*	QDB*	DBQ*	BA	BAN
71	0 1	MSE DA	MSNE DAN	MSLE MSNG DS	MSG DL	MSW DLN	MSNW DLM	MASL DJZ	MASG DTE
72	0 1	EX	SLJ ER	JPS	JNS PAIJ	AH SCN	ANH 1	AT 2	ANT
73	0 1	SSC LSSC	DSC LDSC	SSL LSSL	DSL LDSL	SSA 3	DSA SIL	LSC LCR, LLA	DLSC 4
74	0 1	JZ JNB	JNZ JB	JP JMGI	JN LMJ	J 5	HJ 6	NOP JC	AAIJ JNC
75	0 1	LIC LFC	LICM LFCM	JIC JFC	DIC	LOC AACI	LOCM PACI	JOC ACI	DOC PCI
76	0 1	FA DFA	FAN DFAN	FM DFM	FD DFD	LUF DFU	LCF DFP	MCDU FEL	CDU FCL

*Used only by assembler.

1 LPS, LMP, LUP
2 LSL, LUS, SL, SUL
3 III, EDC, DDC, CES, ES
4 TS, TSS, TCS
5 JO, JFU, JFO, JDF
6 JNO, JNFU, JNFO, JNDF

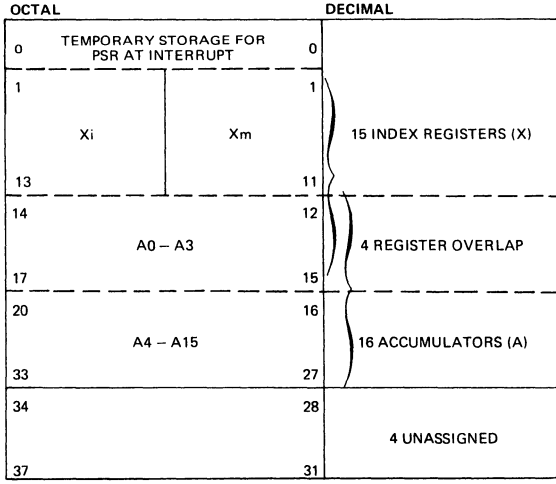
MNEMONIC DESIGNATION AND ABSOLUTE ADDRESSES OF CONTROL REGISTERS AND PARTIAL WORD DESIGNATORS

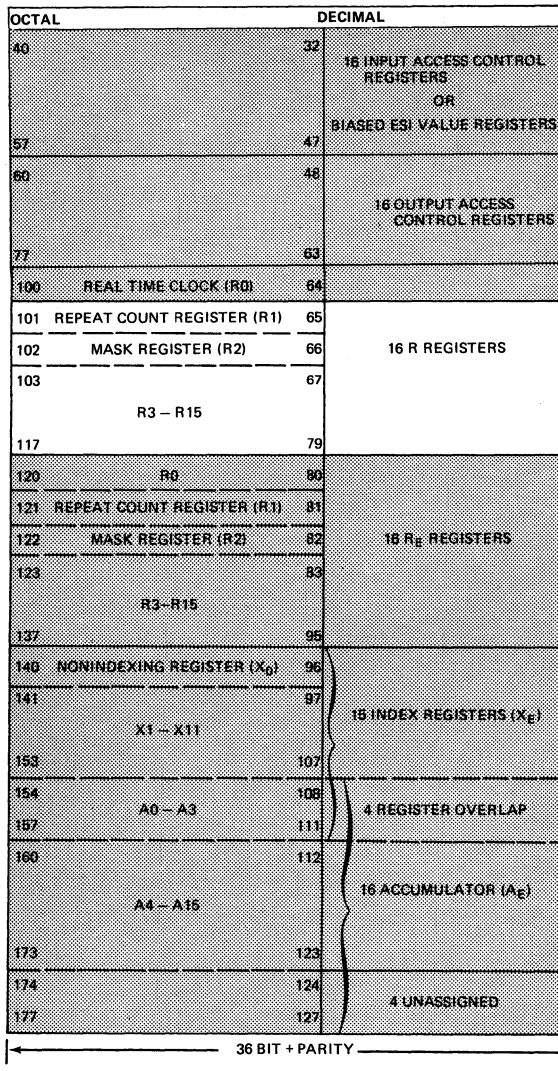
The proc call AXR$ must appear in operand field prior to any references in the code to these mnemonics.

INDEX REGISTERS		ARITHMETIC REGISTERS		R REGISTERS		STAGING REGISTERS J REGISTERS		PARTIAL WORD DESIGNATION	
MNE-MONIC	ABSO-LUTE (OCTAL)	MNE-MONIC	ABSO-LUTE (OCTAL)	MNE-MONIC	ABSO-LUTE (OCTAL)	MNE-MONIC	ABSO-LUTE (OCTAL)	MNE-MONIC	ABSO-LUTE (OCTAL)
X0	0	A0	14	R1	101	SR1	103	W	0
X1	1	A1	15	R2	102	SR2	104	H2	1
X2	2	A2	16	R3	103	SR3	105	H1	2
X3	3	A3	17	R4	104	J0	106	XH2	3
X4	4	A4	20	R5	105	J1	107	XH1	4
X5	5	A5	21	R6	106	J2	110	T3	5
X6	6	A6	22	R7	107	J3	111	T2	6
X7	7	A7	23	R8	110			T1	7
X8	10	A8	24	R9	111			S1	15
X9	11	A9	25	R10	112			S2	14
X10	12	A10	26	R11	113			S3	13
X11	13	A11	27	R12	114			S4	12
		A12	30	R13	115			S5	11
		A13	31	R14	116			S6	10
		A14	32	R15	117			Q1	7
		A15	33					Q2	4
								Q3	6
								Q4	5
								U	16
								XU	17

CONTROL REGISTER ADDRESS ASSIGNMENTS

INTEGRATED CIRCUIT REGISTERS (ICR) UNIVAC 1106/1108





= For use only by the executive (guard mode protected)

- The UNIVAC 1110 GRS is equivalent to the ICRs in the 1106/1108 with the following exceptions:

  (1) The I/O buffer ACWs (addresses 41–77) are moved to the IOAU.

  (2) The values in 40–57 are new.

  (3) The Staging-registers and J-registers are also new (addresses 103–111 and 123–131) and are used for character and byte instruction.

GENERAL REGISTER STACK (GRS)

UNIVAC 1110

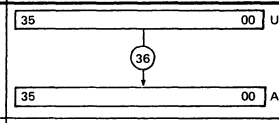
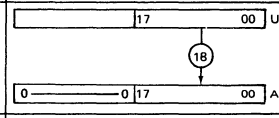
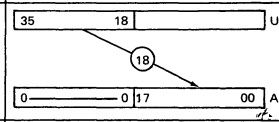
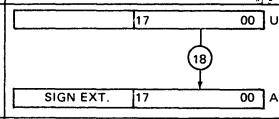
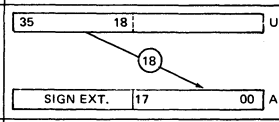
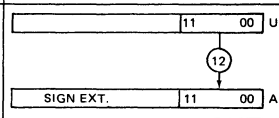
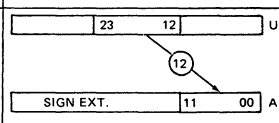
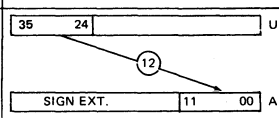
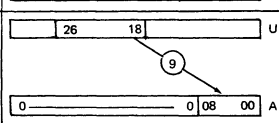
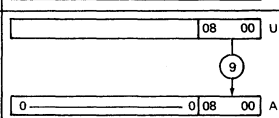
OCTAL		DECIMAL	FUNCTION
0	NON-INDEXING X REGISTER X0	0	
1 – 13	Xi / Xm	1 – 11	16 INDEX REGISTERS (X) X0 – A3
14 – 17	A0 – A3	12 – 15	4 OVERLAPPED (X OR A) A0 – A3
20 – 33	A4 – A15	12 – 27	16 ACCUMULATORS (A) A0 – A15
34 – 37	A15 + 1, A15 + 2 A15 + 3, & A15 + 4 UNASSIGN.	28 – 31	3&4 UNASSIGNED
40 – 41	MAIN PROCESSOR STATE REGISTER TEMPORARY STORAGE	32 – 33	4 PROCESSOR STATE REGISTERS TEMPORARY STORAGE SAVE AREA
42 – 43	UTILITY PROCESSOR STATE REGISTER TEMPORARY STORAGE	34 – 35	
44	EXEC DESCRIPTOR POINTER REG.	36	2 BANK SAVE AREA DESCRIP. POINTER REGS.
45	USER DESCRIPTOR POINTER REG.	37	
46	CURRENT DESCRIPTOR INDICES (PACKED) FOR PSRM	38	2 BANK DESCRIPTOR INDEX REGISTERS
47	CURRENT DESCRIPTOR INDICES (PACKED) FOR PSRU	39	
50	NOT USED	40	UNASSIGNED
51	STORAGE PARITY CHECK STATUS	41	5 INTERRUPT STATUS WORDS SAVE AREA
52	INTERFACE PARITY CHECK STATUS	42	
53	GUARD MODE INTERRUPT STATUS	43	
54	UNDEFINED SEQ. INTERR. STATUS	44	
55	SYSTEM INTERRUPT STATUS	45	
56	PRIMARY STORAGE REF. COUNTER	46	2 STORAGE REFERENCE COUNTERS SAVE AREA
57	EXTENDED STORAGE REF. CNTR.	47	
60 – 77	NOT USABLE	48 – 63	16 NOT USABLE (NO HARDWARE)
100	REAL TIME CLOCK (R0)	64	
101	REPEAT COUNT REGISTER (R1)	65	16 SPECIAL REGISTERS (R)
102	MASK REGISTER (R2)	66	
103 – 105	STAGING REGISTERS SR1 – SR3 R3 – R5	67 – 69	
106 – 111	J–REGISTERS J0 – J3 R6 – R9	70 – 73	
112 – 117	UNASSIGNED R10 – R15	74 – 77	
120	UNASSIGNED R0	80	16 SPECIAL REGISTERS (R)
121	REPEAT COUNT REGISTER R1	81	
122	MASK REGISTER R2	82	
123 – 125	STAGING REGISTERS SR1 – SR3 R3 – R5	83 – 85	
126 – 131	J–REGISTERS J0 – J3 R6 – R9	86 – 89	
132 – 137	UNASSIGNED R10 – R15	90 – 95	
140	NON-INDEXING X REGISTER (X0)	96	16 INDEX REGISTERS (X) X0 – A3
141 – 153	Xi / Xm X1 – X11	97 – 107	
154 – 157	A0 – A3	108 – 111	4 OVERLAPPED (X OR A) A0 – A3
160 – 173	A4 – A15	112 – 123	16 ACCUMULATORS (A) A0 – A15
174 – 177	A15 + 1, A15 + 2 A15 + 3 & A15 + 4 UNASSIGNED	124 – 127	4 UNASSIGNED

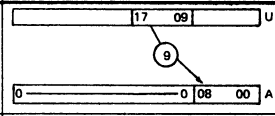
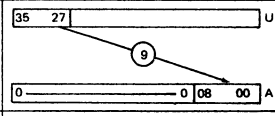
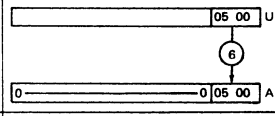
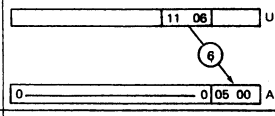
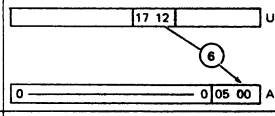
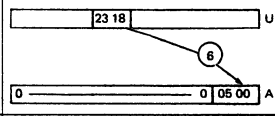
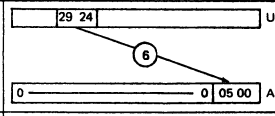
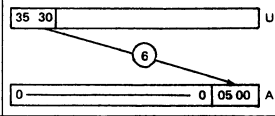
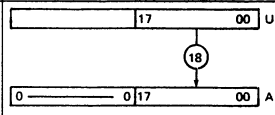
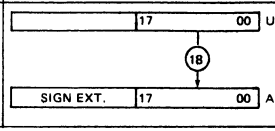
←——36 BIT + PARITY——→

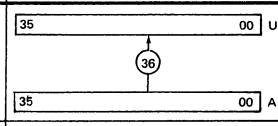
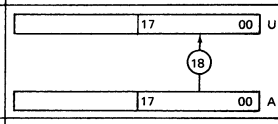
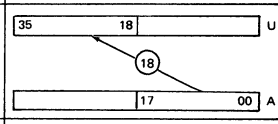
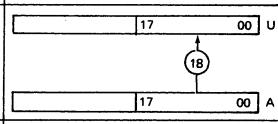
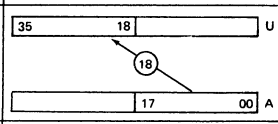
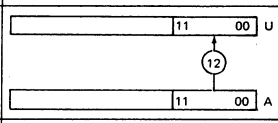
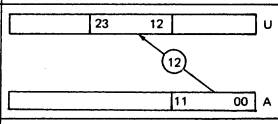
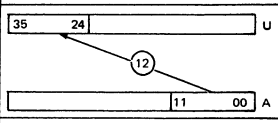
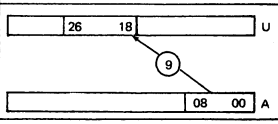
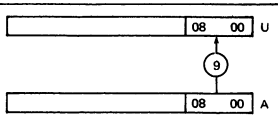
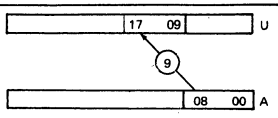
▨ = For use only by the executive (guard mode protected)

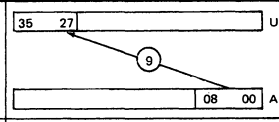
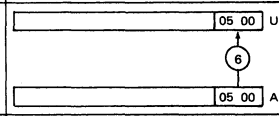
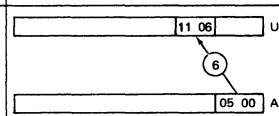
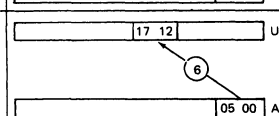
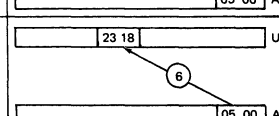
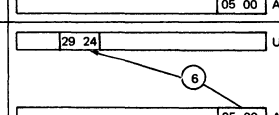
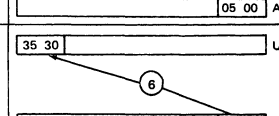
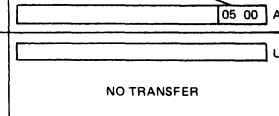
J-DETERMINED PARTIAL WORD TRANSFERS

Note that when a control register is specified in the u field of an instruction, j designators from 1_8 to 15_8 are treated the same as a j designator of 0_8. That is, a whole word register-to-register transfer occurs.

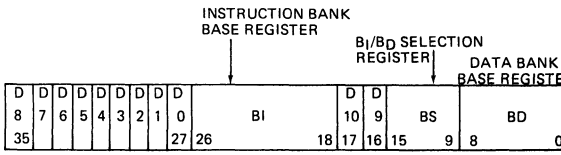
j-Value (Octal)	Quarter Word Designator	Mnemonic for j	Main Storage ↓ Control Register
0	0 or 1	W	
1	0 or 1	H2	
2	0 or 1	H1	
3	0 or 1	XH2	
4	0	XH1	
5	0	T3	
6	0	T2	
7	0	T1	
4	1	Q2	
5	1	Q4	

j-Value (Octal)	Quarter Word Designator	Mnemonic for j	Main Storage ↓ Control Register
6	1	Q3	
7	1	Q1	
10	0 or 1	S6	
11	0 or 1	S5	
12	0 or 1	S4	
13	0 or 1	S3	
14	0 or 1	S2	
15	0 or 1	S1	
16	0 or 1	U	
17	0 or 1	XU	

j-Value (Octal)	Quarter Word Designator	Mnemonic for j	Control Register ↓ Main Storage
0	0 or 1	W	
1	0 or 1	H2	
2	0 or 1	H1	
3	0 or 1	XH2	
4	0	XH1	
5	0	T3	
6	0	T2	
7	0	T1	
4	1	Q2	
5	1	Q4	
6	1	Q3	

j-Value (Octal)	Quarter Word Designator	Mnemonic for j	Control Register ↓ Main Storage
7	1	Q1	
10	0 or 1	S6	
11	0 or 1	S5	
12	0 or 1	S4	
13	0 or 1	S3	
14	0 or 1	S2	
15	0 or 1	S1	
16	0 or 1	U	
17	0 or 1	XU	

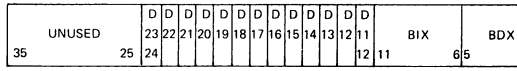
PROCESSOR STATE REGISTER (PSR) FORMAT



D0	Bit 27	Arithmetic Carry Designator
D1	Bit 28	Arithmetic Overflow Designator
D2	Bit 29	Guard Mode
D3	Bit 30	Write Only Storage Protection
D4	Bit 31	(1106/1108) 1107 Compatibility; (1110) Character Addressing Mode

D5	Bit 32	Double Precision Underflow
D6	Bit 33	Control Register Selection
D7	Bit 34	Base Register Suppression if i = 1
D8	Bit 35	Floating-Point Zero
D9	Bit 16	(1106/1108) Not Used; (1110) Index Register Mode 24 Bit if D7=1 and i=1
D10	Bit 17	Quarter Word Mode

PROCESSOR STATE REGISTER EXTENDED* (PSRE) FORMAT

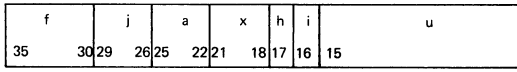


D11	Bit 12	Operand Base Selector PSRU if = 1
D12	Bit 13	PSR/SLR Selector: Utility if = 1
D13	Bit 14	PSRM I Bank Write Inhibit
D14	Bit 15	PSRM D Bank Write Inhibit
D15	Bit 16	PSRU I Bank Write Inhibit
D16	Bit 17	PSRU D Bank Write Inhibit
D17	Bit 18	Residue Store Single Precision FP
D18	Bit 19	PSR/SLR Auto Switch Enable
D19	Bit 20	Allow EXEC Bank Descriptor Pointer
D20	Bit 21	Arithmetic Exception Interrupt Designator
D21	Bit 22	Characteristic Underflow Designator
D22	Bit 23	Characteristic Overflow Designator
D23	Bit 24	Divide Fault Designator

*1110 Only

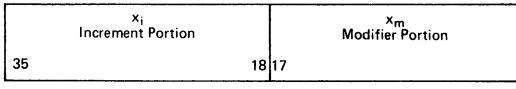
OTHER WORD FORMATS

INSTRUCTION WORD

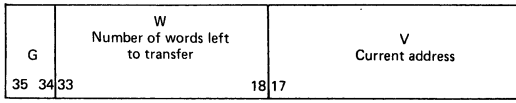


Note that the final operand address generated (specified as bit U), in both direct and indirect addressing, is based on the four fields x, h, i and u.

INDEX REGISTER WORD (Control Registers X1–X11 and A0–A3)



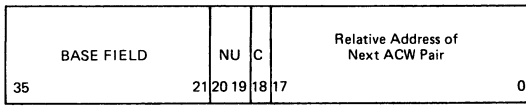
ISI (Internally Specified Index) ACCESS CONTROL WORD



The values of G are 0 – increment, 10_2 – decrement, 1 or 11_2 – neither

*1110 only.

ISI (Internally Specified Index) CHAIN POINTER WORD*

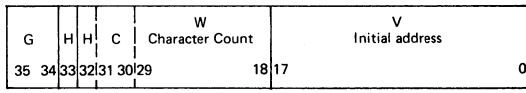


BASE FIELD — The BASE FIELD is added as bits 23–9 to the ACW specified address to form a 24 bit absolute address.

NU — Not used.

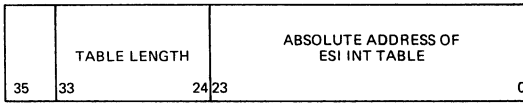
C — Chain bit: if set to one, use address in H2 relative to the chain base register to get the next ACW and CPW.

ESI (Externally Specified Index) ACCESS CONTROL WORD (H is one bit on halfword, two bits on quarterword; C is present for quarterword)

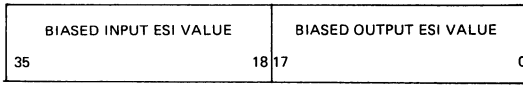


The settings of H toggle back and forth to indicate H2, H1 (or Q4, Q3, Q2, Q1) in successive sequence.

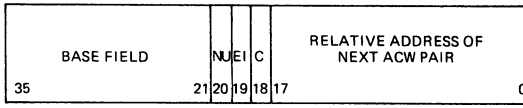
IACW (Input Access Control Word) ACTIVE POINTER *



BIASED ESI VALUES IN IACRs (Input Access Control Registers, 40_8–57_8) – 1106/1108

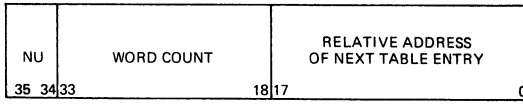


HALF-WORD ESI CHAIN POINTER WORD*

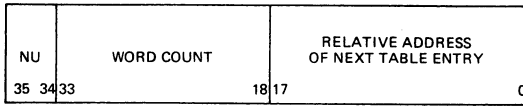


EI CHAIN = When set specifies data chaining when an ESI external interrupt occurs.

ESI ACTIVE POINTER*

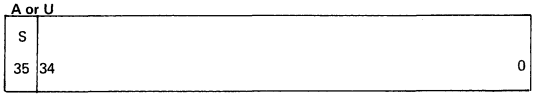


ESI AUXILIARY POINTER*



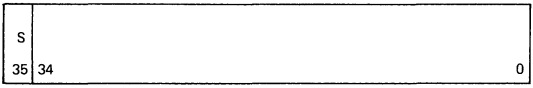
*1110 only.

SINGLE-PRECISION FIXED-POINT WORD (See Fixed-Point Arithmetic Instructions)
A or U



'S' is the sign bit.

DOUBLE-PRECISION FIXED-POINT WORD (See Instruction Functions 71 10–11)
A or U

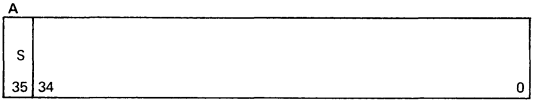


A+1 or U+1

FIXED-POINT MULTIPLY INTEGER (MI) RESULT (See Instruction Functions 30 0–17)
A

A+1

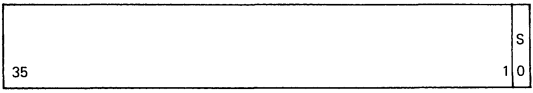
FIXED-POINT MULTIPLY SINGLE INTEGER (MSI) RESULT (See Instruction Functions 31 0–17)
A



Note that the 35 high order (most significant) bits of this result are not retained.

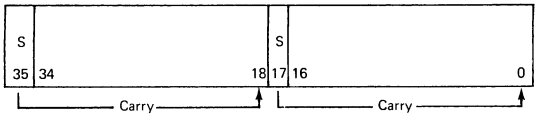
FIXED-POINT MULTIPLY FRACTIONAL (MF) RESULT (See Instruction Functions 32 0–17)
A

A+1

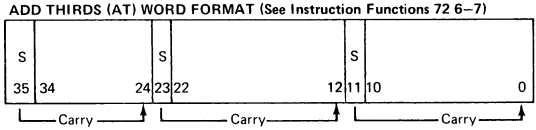


As opposed to the Multiply Integer, above, the Multiply Fractional does a final Left Double Shift Circular of the most high order sign bit, from bit position 35 in A to bit position 0 in A+1.

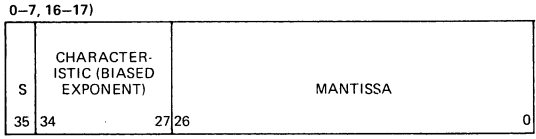
ADD HALVES (AH) WORD FORMAT (See Instruction Functions 72 4–5)



ADD THIRDS (AT) WORD FORMAT (See Instruction Functions 72 6–7)

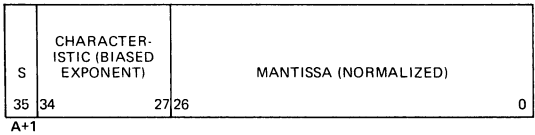


SINGLE-PRECISION FLOATING-POINT OPERAND (See Instruction Functions 76 0–7, 16–17)

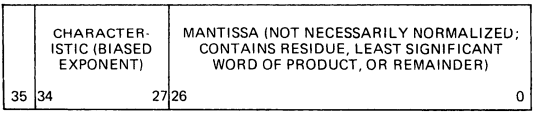


SINGLE-PRECISION FLOATING-POINT RESULT (See Instruction Functions 76 0–7, 16–17)

A

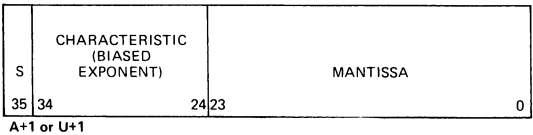


A+1



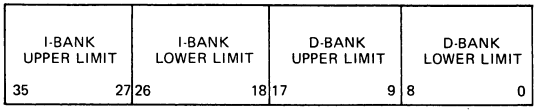
DOUBLE-PRECISION FLOATING-POINT OPERAND OR RESULT (See Instruction Functions 76 10–17)

A or U



A+1 or U+1

STORAGE LIMITS REGISTER (SLR)



These are 9-bit absolute (relative, on the 1110) core block indicators, allowing program addressing ranges up to 262K.

FIXED ADDRESS ASSIGNMENTS

These are all interrupt locations, except the three status words in 200_8–202_8 and the day clock count in 216_8.

DECIMAL ADDRESS	OCTAL ADDRESS	UNIVAC 1106/1108 USE	UNIVAC 1110 USE
128	200	Status word for external interrupt on CPU #0	CAU 0 external interrupt status word IOAU 0
129	201	Status word for external interrupt on CPU #1	CAU 1 external interrupt status word IOAU 0
130	202	Status word for external interrupt on CPU #2	CAU 2 external interrupt status word IOAU 0
131	203	Unassigned	CAU 3 external interrupt status word IOAU 0
132	204	Unassigned	CAU 4 external interrupt status word IOAU 0
133	205	Unassigned	CAU 5 external interrupt status word IOAU 0
134, 135	206, 207	Unassigned	Same
136	210	Power loss interrupt	Same
137	211	ESI access control word parity error interrupt	Jump History Interrupt
138	212	ISI access control word parity error interrupt	IOAU ACR read parity check interrupt (status to 0260–0263)
139	213	I/O data parity error interrupt	IOAU storage parity check interrupt (status to 0260–0263)
140	214	Unassigned	IOAU channel parity interrupt (status to 0260–0263)
141	215	Unassigned	CAU/IOAU interface parity interrupt (status to GRS 052)
142	216	Day clock input	Same
143	217	Day clock interrupt	Same
144	220	ISI input monitor interrupt	Same
145	221	ISI output monitor interrupt	Same
146	222	ISI function monitor interrupt	Same
147	223	ISI external interrupt	Same
148	224	ESI input monitor interrupt	Unassigned
149	225	ESI output monitor interrupt	Unassigned
150	226	Unassigned	ESI table full interrupt
151	227	ESI external interrupt	ESI tabling occurred interrupt
152	230	Status word for external interrupt on unit processor (T #3011–99); unassigned for other CPUs	Undefined sequence interrupt (status to GRS 054)
153	231	Real-time clock interrupt	Same
154	232	Interprocessor interrupt #0	System interrupt (interprocessor) (status to GRS 055)
155	233	Interprocessor interrupt #1	Unassigned

DECIMAL ADDRESS	OCTAL ADDRESS	UNIVAC 1106/1108 USE	UNIVAC 1110 USE
156	234	Unassigned	Storage parity interrupt (status to GRS 051)
157	235	Main storage parity error interrupt (MEM2)	Unassigned
158	236	Main storage parity error interrupt (MEM3)	Unassigned
159	237	Main storage parity error interrupt (MEM4)	Unassigned
160	240	Control register parity error interrupt	GRS parity error interrupt (no status stored)
161	241	Illegal instruction operation (EXEC 8 mnemonic: IOPR) fault interrupt	Illegal instruction interrupt (no status stored)
162	242	Executive Request (ER) interrupt. See instruction function 72 11	Executive Request interrupt (no status stored)
163	243	Guard mode (EXEC 8 mnemonic: IGDM)/storage limits protection fault interrupt	Guard mode fault interrupt (status to GRS 053)
164	244	Test and set (TS) interrupt. See instruction function 73 17	Same
165	245	Floating-point characteristic underflow (EXEC 8 mnemonic: IFUF) interrupt	Same
166	246	Floating-point characteristic overflow (EXEC 8 mnemonic: IFOF) interrupt	Same
167	247	Divide overflow fault interrupt (EXEC 8 mnemonic: IDOF)	Same
168	250	Unassigned	CAU 0 external interrupt status word IOAU 1
169	251	Unassigned	CAU 1 external interrupt status word IOAU 1
170	252	Unassigned	CAU 2 external interrupt status word IOAU 1
171	253	Unassigned	CAU 3 external interrupt status word IOAU 1
172	254	Unassigned	CAU 4 external interrupt status word IOAU 1
173	255	Unassigned	CAU 5 external interrupt status word IOAU 1
174	256	Unassigned	Same
175	257	Unassigned	Same
176	260	Through 377, status words for external interrupts from IOCs	IOAU 0 internal interrupt status word
177	261		IOAU 1 internal interrupt status word
178	262		IOAU 2 internal interrupt status word
179	263		IOAU 3 internal interrupt status word
180	264		Breakpoint interrupt
Last Address–1		Main storage parity error interrupt (MEM 1)	Parity check interrupt memory 1 (status to GRS 051)

HARDWARE STATUS CODES

BIT POSITION	UNISERVO VI-C/VIII-C MAGNETIC TAPE SUBSYSTEM
35–18	UNASSIGNED
17	ABNORMAL FRAME COUNT
16	(POSSIBLE) END OF FILE
15	END-OF-TAPE WARNING/LOAD POINT
14	INTERLOCK
13	BUSY
12	PARITY ERROR
11	LATE ACKNOWLEDGE ERROR
10	TAPE HASH
9	INVALID FUNCTION
8–5	BAD TRACK
4	NORMALLY A ONE BIT; A ZERO BIT FOR CHANNEL PARITY (1110 ONLY)
3–0	FRAME COUNT/MODULO 6 (FOR 7 TRACK) OR MODULO 9 (FOR 9 TRACK)

UNISERVO 12/16/20 SENSE DATE BYTES

SENSE DATA BYTE	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
0	COM-MAND REJECT	INTER-VENTION RE-QUIRED	BUS OUT CHECK	EQUIP-MENT CHECK	DATA CHECK	OVER-RUN	WORD COUNT ZERO	DATA CON-VERTER CHECK
1	NOISE	TAPE UNIT STATUS A*	TAPE UNIT STATUS B*	7-TRACK*	LOAD POINT*	END-OF-TAPE*	FILE PROTECT*	TAPE UNIT IN-COMPAT-IBILITY
2	TRACK IN ERROR							
3	R/W VRC	MDT CHECK TRACK START FAILURE/ LRC	SKEW	POST-AMBLE CHECK/ CRC	W/VRC DEAD TRACK	TAPE UNIT 1600 BPI*	BACK-WARD*	NOT USED; ALWAYS 0
4	RUN-AWAY CHECK	TAPE MOTION FAULT	SPEED CHECK (UNI-SERVO 20 ONLY)	TEST / ALWAYS 0 BITS	STALL	TAPE FAULT		TEST

NOTE:

Asterisk (*) indicates a bit that is conditioned by current status of tape unit.

Legend:

CRC – cyclic redundancy check

LRC – longitudinal redundancy check

R/W – read/write

VRC – vertical redundancy check

CODE (OCTAL)	FH-432/1782 (T5012 CU), FH-432 (T6013 CU), AND FH-880
02	LATE ACKNOWLEDGE (T5012 CU ONLY)
04	END OF BLOCK
05	SEARCH FIND
06	OVERFLOW PARITY ERROR
07	NONCONTINUOUS READ PARITY ERROR (EXCEPT FOR T5012 CU). ALSO CHARACTER COUNT ERROR (FH-880 ONLY)
14	FAULT
15	CHANNEL PARITY (T5012/1110 ONLY)
20	ANGULAR ADDRESS (T5012 ONLY)
30	SYNCHRONIZER CHARACTER COUNT ERROR (FH-880 ONLY)
34	END-OF-FILE
40	NORMAL COMPLETION
50	ILLEGAL FUNCTION
54	ILLEGAL ADDRESS

CODE (OCTAL)	FH-432/1782 (T5012 CU), FH-432 (T6013 CU), AND FH-880
60	CONTROL SEQUENCE ERROR (FH-880 ONLY)
6X	PARITY ERROR (T5012 CU ONLY)
64	CONTINUOUS READ PARITY ERROR (EXCEPT FOR T5012 CU)
70	WRITE CHARACTER COUNT ERROR (FH-880 ONLY)

CODE (OCTAL)	FASTRAND/8460 MASS STORAGE SUBSYSTEM
04	TIME OUT (INPUT)
05	END-OF-POSITION (INPUT)
06	PHASE CHECK ERROR
07	NONRECONSTRUCTIBLE ERROR
10	DATA RECOVERED
14	ADDRESS ERROR
20	LATE ACKNOWLEDGE
24	END-OF-POSITION (OUTPUT)
34	SECTOR LENGTH ERROR
40	NORMAL COMPLETION
50	INVALID FUNCTION CODE
54	NO RESPONSE/WRITE LOCKOUT
60	WRITER ERROR
61	CHANNEL PARITY ON DATA (1110 ONLY)
71	CHANNEL PARITY ON EXTERNAL FUNCTION (1110 ONLY)

CODE (OCTAL)	PUNCHED CARD SUBSYSTEM
20	SYNCHRONIZER SEQUENCE ERROR
30	SYNCHRONIZER COUNTER ERROR
40	NORMAL COMPLETION
50	ILLEGAL FUNCTION
54	READ/PUNCH CHECK ERROR
60	INAPPROPRIATE FUNCTION CODE
70	ILLEGAL CHARACTER CODE
74	INTERLOCK FAULT

CODE (OCTAL)	HIGH-SPEED PRINTER SUBSYSTEM (T7299 CU OR T5011 CU)
40	NORMAL COMPLETION
44	OUT OF FORMS
50	INVALID FUNCTION
54	PRIORITY CONFLICT (T5011 CU ONLY)
60	SHARED PATH AVAILABLE (T5011 CU ONLY)
70	MEMORY COMPARE ERROR (T5011 CU ONLY)
74	INTERLOCK FAULT

DISC HARDWARE STATUS CODES

STATUS AND FLAG MASK BYTE

BIT	0	1	2	3	4	5	6	7
STATUS BYTE	ATTEN-TION	STATUS MODIFIER	CONTROL UNIT END	BUSY	CHANNEL END	DEVICE END	UNIT CHECK	UNIT EXCEP-TION
FLAG BYTE	Odd No. Record	Overflow Record (Not Last)	"0"	"0"	"0"	"0"	Defective Track	Alternate Track

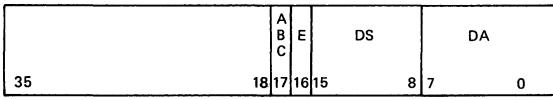
8414/8424 SENSE DATA BYTE FORMAT

SENSE DATA BYTE	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
0	COM-MAND REJECT	INTER-VENTION RE-QUIRED	BUS OUT CHECK	EQUIP-MENT CHECK	DATA CHECK	OVER-RUN	TRACK CONDI-TION CHECK	SEEK CHECK
1	COUNT AREA CHECK	TRACK OVER-RUN	CYLIN-DER END	INVALID SE-QUENCE	NO RECORD FOUND	FILE PRO-TECTED	MISSING ADDRESS MARKER	OVER-FLOW INCOM-PLETE
2	UNSAFE 0)	NOT USED (ALWAYS 0)	NOT USED (ALWAYS 0)	NOT USED (ALWAYS 0)	NOT USED (ALWAYS 0)	UNSE-LECTED STATUS	NOT USED (ALWAYS 0)	NOT USED (ALWAYS 0)
3	READY	ONLINE	UNSAFE	NOT USED (ALWAYS 0)	NOT USED (ALWAYS 0)	END OF CYLIN-DER	NOT USED (ALWAYS 0)	SEEK INCOM-PLETE
4	ALWAYS 0							
5	THIS BYTE IS ALL 0'S EXCEPT WHEN BIT 7 OF BYTE 1 IS SET (OVERFLOW INCOMPLETE).							

8440 SENSE DATA BYTE FORMAT

SENSE DATA BYTE	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
0	COM-MAND REJECT	INTER-VENTION RE-QUIRED	BUS OUT CHECK	EQUIP-MENT CHECK	DATA CHECK	OVER-RUN	DEFEC-TIVE TRACK CHECK	SEEK CHECK
1	COUNT AREA CHECK	TRACK OVER-RUN	END OF CYLIN-DER	INVALID SE-QUENCE	NO RECORD FOUND	FILE PRO-TECTED	MISSING ADDRESS MARKER	OVER-FLOW INCOM-PLETE
2	DEVICE MAL-FUNC-TION	ECC CHECK	NO DEVICE CLOCKS RECEIVED	QUEUING PARITY	PRIORITY TER-MINATE	UNSE-LECTED STATUS	SECTOR ERROR	ECC PARITY ERROR
3	READY	ONLINE	UNSAFE	KEY ECC CHECK	ONLINE	END OF CYLIN-DER-DEVICE		SEEK INCOM-PLETE
4	ALWAYS 0							
5	THIS BYTE IS 0'S EXCEPT WHEN BIT 7 OF BYTE 1 IS SET (OVERFLOW COMPLETE).							

MSA STATUS WORD FORMAT



ABC		Bit 17	Abnormal Byte Count
E		Bit 16	Error Condition Detected By MSA
DS			Device Status Byte
	ATT	Bit 15	Attention
	SM	Bit 14	Status Modifier
	CUE	Bit 13	Control Unit End
	B	Bit 12	Busy
	CE	Bit 11	Channel End
	DE	Bit 10	Device End
	UC	Bit 9	Unit Check
	UE	Bit 8	Unit Exception
DA		Bits 7–0	Device Address Byte

NOTE:

Shaded bit positions are unused and should be zero-filled.

MSA AUXILIARY STATUS WORD FORMAT

		NOP	IPE	TCK	CLE	ACP	XCK	LA	SC	WPC	IS	IC	FA		M	
35	18	17	16	15	14	13	12	11	10	9	8	7	6	4	3	0

NOTE:

Conditions setting bit positions 7-17 cause bit position 16 of the normal status word to be set. Shaded bit positions are unused and zero-filled.

NOP	Bit 17	Control Unit Nonoperational
IPE	Bit 16	Input Parity Error
TCK	Bit 15	Time Check
CLE	Bit 14	Control Line Error
ACP	Bit 13	Address Compare Error
XCK	Bit 12	Translate Check
LA	Bit 11	Late Acknowledge
SC	Bit 10	Stall Check
WPC	Bit 9	Word Parity Check (Channel Parity)
IS	Bit 8	Invalid Sequence
IC	Bit 7	Invalid Command
FA	Bits 6-4	Function Address
M	Bits 3-0	Magnitude of Byte Count

POSITIVE AND NEGATIVE POWERS OF 2 (BINARY), 8 (OCTAL) AND 16 (HEXADECIMAL)

POWER OF			DECIMAL VALUE
2	8	16	
0	0	0	1
1			2
2			4
3	1		8
4		1	16
5			32
6	2		64
7			128
8		2	256
9	3		512
10			1024
11			2048
12	4	3	4096
13			8192
14			16384
15	5		32768
16		4	65536
17			1 31072
18	6		2 62144
19			5 24288
20		5	10 48576
21	7		20 97152
22			41 94304
23			83 88608
24	8	6	167 77216
25			335 54432
26			671 08864
27	9		1342 17728
28		7	2684 35456
29			5368 70912
30	10		10737 41824
31			21474 83648
32		8	42949 67296
33	11		85899 34592
34			1 71798 69184
35			3 43597 38368
36	12	9	6 87194 76736

POWER OF			DECIMAL VALUE
2	8	16	
−1			.5
−2			.25
−3	−1		.125
−4		−1	.062 5
−5			.031 25
−6	−2		.015 625
−7			.007 812 5
−8		−2	.003 906 25
−9	−3		.001 953 125
−10			.000 976 562 5
−11			.000 488 281 25
−12	−4	−3	.000 244 140 625
−13			.000 122 070 312 5
−14			.000 061 035 156 25
−15	−5		.000 030 517 578 125
−16		−4	.000 015 258 789 062 5
−17			.000 007 629 394 531 25
−18	−6		.000 003 814 697 265 625
−19			.000 001 907 348 632 812 5
−20		−5	.000 000 953 674 316 406 25
−21	−7		.000 000 476 837 158 203 125
−22			.000 000 238 418 579 101 562 5
−23			.000 000 119 209 289 550 781 25
−24	−8	−6	.000 000 059 604 644 775 390 625
−25			.000 000 029 802 322 387 695 312 5
−26			.000 000 014 901 161 193 847 656 25
−27	−9		.000 000 007 450 580 596 923 828 125
−28		−7	.000 000 003 725 290 298 461 914 062 5
−29			.000 000 001 862 645 149 230 957 031 25
−30	−10		.000 000 000 931 322 574 615 478 515 625
−31			.000 000 000 465 661 287 307 739 257 812 5
−32		−8	.000 000 000 232 830 643 653 869 628 906 25
−33	−11		.000 000 000 116 415 321 826 934 814 453 125
−34			.000 000 000 058 207 660 913 467 407 226 562 5
−35			.000 000 000 029 103 830 456 733 703 613 281 25
−36	−12	−9	.000 000 000 014 551 915 228 366 851 806 640 625