ComputerAutomation

NAKED MINI. Division

18651 Von Karman, Irvine, California 92713 Tel 714 833 8830 TWX 910 595 1767

CAI Limited
Hertford House, Denham Way, Rickmansworth, Herts WD3 2XD
TEL RICKMANSWORTH 71211 • TELEX 922654

OPERATING SYSTEM

USER'S MANUAL

96530-00D5 April 1976

PRINTED IN THE U.S.A.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

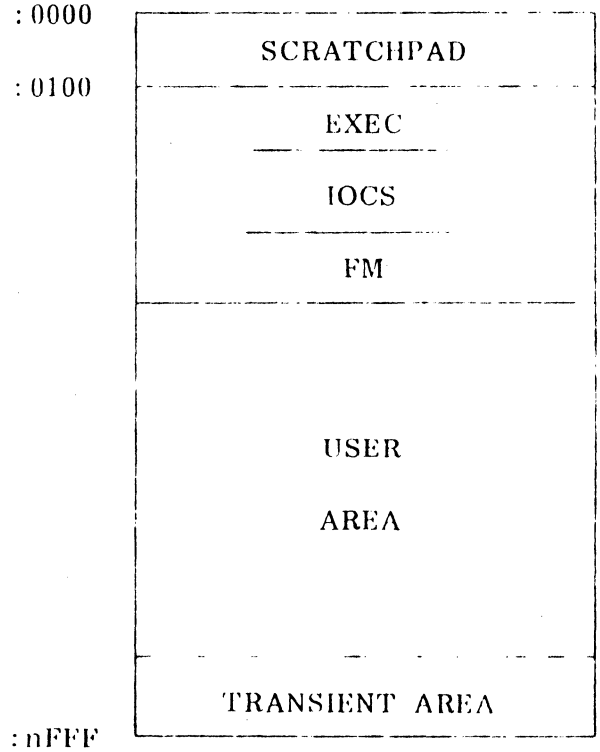
Section 1

THE CAI OPERATING SYSTEM

1.1 STRUCTURE OF THE SYSTEM

The CAI Operating System provides the tools required for efficient program development and execution, in both batch and on-line modes. The minimum configuration for OS is one ALPHA/LSI (or ALPHA-16) processor with 16K words of memory, one ASR-33 Teletype, and one magnetic peripheral device. The device selected for loading OS itself determines whether the software environment is termed Disk Operating System (DOS), Magnetic Tape Operating System (MTOS), or Cassette Operating System (COS). User programs may reside on any of these devices, or all of them, and on paper tape as well.

As loaded into memory, OS consists of the Executive (EXEC), thru which the user controls the entire system with a command language, the Input/Output Control System (IOCS), which "drives" the peripherals, and the File Manager (FM), which provides access by name to files on magnetic recording media. During the execution of certain System Utility Programs, a small portion of high memory, called the Transient Area, will temporarily be reserved. The remainder of memory (including high memory if the Transient Area is not actually in use) is called the User Area.



OS Memory Usage

Section 2

OPERATOR/SYSTEM COMMUNICATION

2.1 INTRODUCTION

The operator console (a teletype or other interactive device) is the basic communication medium between the user and the operating system. Through this console, the user communicates with the system executive routines, user programs operating under the system, and system utilities.

This is not, however, the only such interface. Other devices, such as a card reader or line printer may alternatively be assigned as the system command input (CI) and command output (CO) devices. Such device assignments allow unattended system operation in a batch mode.

2.2 THE OPERATOR CONSOLE

The operator console is the standard command input (CI) and command output (CO) device for the operating system. Although other command I/O assignments may be made, the system will revert to these initial assignments between jobs (/JOb directive) and when a console interrupt is processed.

The console may also serve as a normal system input or output device under OS, and in this mode it can be considered like any other assignable serial device.

When performing input (for either the system or user program) certain keys on the console keyboard have special functions.

1. RETURN. The RETURN key indicates the end of a line of input and causes a carriage return and line feed to be generated.

2. BACKARROW (←). The backarrow causes the previous character input to be replaced by the next character typed. Multiple characters may be replaced by typing the appropriate number of backarrows followed by the correction characters.

3. BACKARROW (←) /RETURN. A backarrow followed immediately by RETURN causes the entire current line to be ignored and replaced by the next line input. The RETURN causes a carriage return and line feed to be generated.

The system indicates to the user that a line of input is required by issuing one of two query characters. The greater than (>) character is printed when the system requires a system command, whereas the question mark (?) character is printed when the system or user program requires parameter or data input. If necessary, the operator may answer "?" with a system command. After OS has processed the command, another "?" will come up, because the original need for an appropriate response still exists.

2.3 ALTERNATE SYSTEM COMMUNICATION

A device other than the console may be used for command input (CI) and/or command output (CO) devices. This can be accomplished by use of the /ASsign or /BAtch commands and will stay in effect until another re-assignment, the start of the next job, or the recognition of a console interrupt.

When the command input (CI) device is assigned to an alternate, all system commands will be echoed (printed) on the command output (CO) device to provide the user with a documented history of the operations performed. Neither of the query characters (> or ?) are printed during this mode of operation.

2.4 CONSOLE INTERRUPT

The system, or any user program operating under it, may be interrupted by use of the console interrupt feature. This interrupt is caused by momentary operation of the INT switch on the computer (AUTO on an ALPHA-16) when in RUN mode.

This interrupt causes the currently executing program (user or system) to be halted at its current position in a resumable manner. The program's current status is saved, the command input and output devices are automatically reassigned (as required) to the operator console and the system requests further action from the operator.

The interrupt does not take place immediately, but allows a small period of time for outstanding I/O to complete. Should this time be exceeded, the system will take control under the assumption that the I/O cannot be completed normally, and a /CAncel function will be simulated.

The status of the executing program at the time of the interrupt may be displayed by use of the /STatus command, and the program may be continued with a /REsume command.

If a /REsume is not given a /CAncel or /JOb should be given to assure proper file management table status.

2.5 SYSTEM COMMANDS

The system commands are the means by which the user communicates with the operating system and controls its actions and operations. Thus, the operator can load programs into memory, start and stop them, dump them to a storage medium, and perform all other required operations.

A system command is distinguished by a slash (/) character in its first character position, immediately followed by the command keyword and zero or more operands. A system command consists of a single logical record (line) on the command input (CI) device; no continuation is allowed.

A keyword is a sequence of letters having special significance to the system. All letters after the first two are optional and may be included or omitted at the user's discretion.

The optional operand field is separated from the command name by one or more blanks and contains zero or more command arguments, separated by commas, without imbedded blanks. The format and number of such arguments are wholly dependent on the individual command. After at least one space, user comments may be added.

The commands acceptable to the operating system are described in detail in the remainder of this section. The following conventions apply:

1. Command keywords are shown beginning with a slash (/) character, followed by the first two letters of the keyword in capitals. The rest of the word is in lower case, signifying that these characters are optional.

2. Square brackets [ ] enclose operands or parameters which are optional and may be included or omitted at the user's discretion.

3. A right square bracket followed by an ellipsis (]...), indicates that the enclosed element may be omitted or repeated an arbitrary number of times.

4. Braces { } indicate that a choice must be made from the enclosed elements.

5. System output is underlined to distinguish it from user input.

System commands can be rejected for a variety of reasons. Any such rejection will cause the system to reassign the command input (CI) and command output (CO) units to the operator console. The system will then print the appropriate error message and pause for remedial operator action.

A general cause of command rejection will be invalid or illegal command formats or parameters. This type of error will cause the message "*CMND REJECT" to be displayed and the system to request a new command.

Assignment of the command input or output unit to an inoperable device may cause a processor stop. Refer to Section 2.6 for details.

2.5.1 The /ASsign Command

$$\text{/ASsign} \quad \text{unit} = \left\{ \begin{matrix} \text{unit} \\ \text{device} \end{matrix} \right\} \left[,\text{unit} = \left\{ \begin{matrix} \text{unit} \\ \text{device} \end{matrix} \right\} \right] \ldots$$

The /ASsign command assigns a logical unit to a physical device, or to another logical unit. It can be given at any time and supersedes any prior assignment of the logical unit.

A unit is defined as any of the two-character symbolic logical unit names described in Appendix D, and denotes a mode of I/O operation.

A device is defined as any of the two-character symbolic physical device names shown in Appendix C and supported at this installation, and denotes the physical device on which the logical operation is to be performed. Examples of various uses of the /ASSIGN command are discussed below.

(1) /ASSIGN SI=CR

In the first example, the command tells the system that henceforth all programs and operations which require symbolic input will receive that input from the card reader. Note that any previous assignments of SI are now lost, but previous assignments to CR are still valid. It is thus possible to share a physical device among several logical units, as shown in example (2).

(2) /ASSIGN CI=CR,SI=CR

In some cases, the physical device assigned supports multiple files, by name. This is the case for magnetic tapes, cassettes, and disks. For these devices, the system requires that the file 'name' be available at the time the file is 'OPENed' for reading or writing. One means of supplying this name is through the /ASsign command by appending it to the physical device symbolic name, separated by a period. This is shown in example (3).

(3) /ASSIGN BO=M1.TEST1

Such an explicit file name assignment always supersedes any previous or subsequent file name definition stored in the program's file control block (FCB). A file name may be assigned to a non-bulk device, but has no effect on OS operation.

(4) /ASSIGN SI=CR,SA=SI

Example (4) demonstrates assignment of a logical unit to another logical unit. This command causes both SI and SA to be assigned to the card reader (CR). A subsequent assignment of

/ASSIGN SI=PR

causes SI to be reassigned to the high speed paper tape reader (PR), while leaving SA still assigned to the card reader.

(5) /ASSIGN SI=CD
*CD NOT FOUND

As described above, the unit and device fields of the /ASsign command are required to be one of several standard symbolic names. Should the operator enter a name which is not among this group, or is not supported at this installation, the system will reject the command as in example (5).

(6) /ASSIGN BI=PR,SI=CD,BO=PP
*CD NOT FOUND

Note that an error on an assignment command with multiple entries will cause all entries after the erroneous one to be ignored. As in example (6), the "BI" unit was assigned to the "PR" device, but because "CD" was invalid, the "BO" assignment was not made.

Other assignment errors, such as failure to assign a valid logical or physical unit, or to provide a file name when required, are diagnosed when the file is OPENed. These conditions will cause error messages to be produced and the system will pause to allow remedial action by the operator.

(7) *S2 UNASSIGNED
>/ASSIGN S2=D0.TEST
>/RESUME

Assume in example (7), the operator failed to assign the "S2" unit (and a default system assignment does not exist). This would cause the error message shown when an attempt to 'OPEN' the device was made, and a system query for remedial action. The operator can then provide the required assignment and RESUME operation.

(8) *SI FILE NAME?
>/ASSIGN SI=M1.TEST
>/RESUME

Likewise, example (8) indicates that a file-oriented 'OPEN' was attempted and the file name was not specified, either in the user's file control block (FCB) or at the initial assignment. Again the operator can correct the situation and continue operation.

2.5.2 The /BAtch Command

/BAtch device

The /BAtch command is a shorthand method of reassigning the command input (CI) logical unit to a new physical device. It performs the same function as an explicit assignment of the command input unit to the device specified using the /ASsign command.

As in the /ASsign command, the device may be any of the two character physical device symbolic names shown in Appendix C.

(1) /BATCH CR

In example (1) above, the command tells the system that henceforth, all commands will be expected to come from the card reader.

(2) /BATCH M0.APROCS

Should the batch device be file oriented, the file name can be appended in the same manner as the /ASsign command. This is illustrated in example (2).

(3) /BATCH CD
*CD NOT FOUND

As in the /ASsign command, the device is required to be one of the several standard physical device names. Should the name entered not be among this group, the command will be rejected as in example (3).

(4) /BATCH M0
*NOT FOUND
>/BATCH M0.APROCS

Failure to provide a file name when required will cause an error message and a system request for remedial action, as in example (4).

2.5.3 The /BEgin Command

/BEgin [address][,parameters]...

The /BEgin command allows the user to start (or restart) a program already in memory. The program must be in an operational state: loaded or terminated, but not cancelled or suspended.

(1) /BEGIN

The command may be given without a starting address, as in example (1). In this case, the starting address given at load time is used. If no address was available at load time, the command will be rejected.

(2) /BEGIN 14A0

As shown in example (2), the command may also be given with a hexadecimal starting address. This address must be higher in memory than the area reserved for OS itself, or the command is rejected. It has precedence over any load time start address. It is, however, a one-time address and does not replace the load time address.

(3) /BEGIN 14A0,YES

(4) /BEGIN ,YES

It is possible to pass parameters to the program being started, just as in the /EXecute command (see Section 2.5.7). If parameters are given, the start address or a leading comma (,) is required, as illustrated in examples (3) and (4).

2.5.4 The /CAncel Command

/CAncel

This command terminates execution of the current program. It does not save any program registers and does not leave the program in a restartable state.

A /CAncel command is normally given when the operator determines a program is not executing properly and must be terminated. /CAncel causes all I/O to terminate immediately, and I/O operations will not come to their logical conclusion. Any output file not already closed with the "keep" option will be unsuitable for future use.

A /CAncel will also be performed, under certain circumstances, when the console interrupt switch is activated (see Console Interrupt description). Also, a /JOb command always performs a /CAncel.

2.5.5 The /COmment Command

/COmment [text]

This command is commentary only and causes no system activity. The remainder of the line after the command field is available for whatever comments the user desires, and will be printed on the command output (CO) device.

2.5.6 The /DAte Command

/DAte [aa/bb/cc]

This command allows the user to display and/or set the system date, which is then available to system and user programs and is displayed by certain system commands. The date is not automatically advanced at 24:00 and must be reset daily.

(1) /DATE
*07/04/72

The command may be given without a parameter, as in example (1). In this case, the user is asking that the current system date be displayed.

(2) /DATE 9/15/72
*09/15/72

The command may also be given with a parameter, and the system will reset its date to this value. The system will again display the date to indicate acceptance, as in example (2).

The parameter shown as "cc" must be exactly 2 alphanumeric characters. In contrast, "aa" and "bb" may each be 1 or 2 characters; the system will supply a leading zero if only one character is entered. The following would be perfectly valid:

(3) /DATE 1/JA/76
*01/JA/76

2.5.7 The /EXecute Command

/EXecute program[,parameter][,parameter]...

This command causes the loading and execution of a program from the System File (SF) unit. The program must be in absolute or relocatable format, and the proposed load must not overlap the area reserved for the Operating System itself.

If a program file contains Loader Type Codes equivalent to any of the assembler (or compiler) facilities listed here, it can not be brought into memory with /EXEC. It must be processed into acceptable format with OS:LDR or OS:LNK.

--Directives which create external references (EXTR, SEXT, REF, SREF, LOAD)
--Directives which create named entry points (NAM, SNAM)
--Directives which contribute to a load-time structure (CHAN)
--Directives which allocate relocatable scratchpad (SREL)
--References, explicit or implicit, to literal values in scratchpad

(1) /ASSIGN SF=M0
/EXEC TEST1

The program to be loaded is specified in the first field following the /EXecute command, and becomes the index to the system file (SF) directories. In example (1) magnetic tape unit 0 is assigned as the system file unit and the system is requested to load and execute a program located on it called TEST1.

(2) /ASSIGN SF=PR
/EXEC DUMMY

If, however, the system file unit is assigned to a non-directoried device (such as paper tape) the next program found on that device is loaded, regardless of its name. Example (2) indicates that condition.

With the exception of those OS utilities which reside in the transient area (OS:LDR, OS:DMP, OS:DBG), a program loaded with the /EXecute command will overlay (and destroy) any previous programs resident in memory.

(3) /ASSIGN SF=M0
/EXEC TEST1,4,6

The optional parameter field allows any number of arguments to be passed to the program after execution has begun (line length cannot exceed 80 characters). These arguments are saved by the system and made available to the executing program by use of the SUPV: call. Their order and format will be strictly a function of the executing program.

(4) /ASSIGN SF=M0
/EXEC TEST,WEEKLY,3
*TEST NOT FOUND

Should the system be unable to locate the program specified on the "SF" device, the command will be rejected as shown in example (4) above.

Should an error occur while attempting to load the requested program, loading will terminate with one of the following messages:

MEM FULL — The system has determined there is insufficient memory (scratchpad or main) available to complete the load.

BAD TC — Invalid code for /EXEC or /LOAD processing.

I/O ERR — An unrecoverable I/O error occurred.

2.5.8 The /JOb Command

/JOb comments

This command indicates to the system that a new sequence of operations is to begin, possibly by a new user, and that all system variables are to be restored to their initial values. This command is available to simplify the system reassignment after a previous step or user. The remainder of the command line is available for commentary.

System reassignment includes the reassignment of all logical I/O units to their initial physical devices (as defined at system generation) and the resetting of all scratchpad and main memory core variables to their initial (minimum/maximum) values. This makes available the maximum system resources to the next job step. In addition, a /JOb command causes a /CAncel function to be performed. Thus completion of a /JOb function may cause a delay of up to three seconds.

Under DOS, the /JOb command must be given whenever the operator loads a different removable disk platter, in order to reset the disk directory information maintained in memory by the disk file manager.

```
/JOB
*09/14/72
*10:15:07
```

The system responds to the /JOb command with the current date and time on the command output (CO) device. The current time is also saved for later display by the /NJob command (see Section 2.5.11).

2.5.9 The /LIst Command

/LIst [logical unit name]

This command displays on the command output device (CO) the current assignments of logical units to physical devices.

(1) /LIST

*CI	TK
*CO	TY
*SI	CR
*LO	LP
*BI	PR
*BO	D1.NAME
*S3	--
*S4	--

For each unit, or for one specified unit, the first column contains the logical unit name, and the second contains the name of the physical device to which it is currently assigned and the associated file name (if any).

2.5.10 The /LOad Command

/LOad program

The /LOad command is similar to the /EXecute Command, and has the same restrictions. After loading, control returns to the system instead of to the loaded program. Once loaded, the program may be entered for execution by using the /BEgin command with any parameters required by the program.

The /LOad command provides a convenient method of debugging a program with OS:DBG. For example, suppose that a user's program, named TEST1, resides on disk unit 1, and requires a correction or "patch" before it can be run. Suppose further that TEST1 requires two parameters (PRAM1 and PRAM2):

```
/AS  SF=D1
/LO  TEST1
/AS  SF=D0  OS:DBG IS ON D0
/EX  OS:DBG
```

At this point, TEST1 has been loaded, and OS:DBG has been entered. Relocation register 0 in OS:DBG (R0) is automatically set to the first location of TEST1. The correction can now be made through register R0, after which the following commands may be entered:

```
T  (to terminate OS:DBG)
/BE  ,PRAM1,PRAM2  (to begin TEST1)
```

Possible load error messages described in the /EXecute command also apply to /LOad.

The LOad command should not be used to load those utility programs which reside in the Transient area of memory (OS:LDR, OS:DMP, OS:DBG). If this is attempted, the utility will be loaded, but a subsequent /BEgin command will not be accepted by the system.

2.5.11 The /NJob Command

```
/NJob  comments
```

This command is largely documentary and indicates to the system that a logical sequence of operations has been completed since the previous /JOb or /NJob command. This command is available to delimit and document steps within the user's job stream. The remainder of the line is available for commentary.

```
/JOB
*09/14/72   10:15:07
/EXEC  TEST
/NJOB
*10:15:07   10:20:35
/BEGIN
/NJOB
*10:20:35   10:25:07
```

The system responds to the /NJob command with the last /JOb or /NJob time and the current time on the command output (CO) device.

2.5.12 The /REsume Command

```
/REsume [parameters]...
```

The /REsume command allows the operator to continue execution of a core-resident program which suspended itself or was suspended by the system in response to a console interrupt. A program which was cancelled or has terminated itself is not resumable.

When a program suspends itself, or a console interrupt occurs, the system restores the command input (CI) and command output (CO) devices to their initial assignments (generally the operator console), allowing whatever operator action is required to take place. Execution of the /REsume command restores these assignments to their previous values.

The system responds to the /REsume command by printing the current time.

```
(1) /RESUME
    *14:07:15
```

Example (1) illustrates the general case where a suspended or interrupted program is resumed. Execution will continue at the next logical instruction with all program registers and status restored.

```
(2) /RESUME YES
    *14:07:15
```

It is possible to pass parameters to the program being resumed, just as in the /EXecute command. This is illustrated in example (2) and would generally apply to a program which suspended (SPND:) itself for operator action and/or response.

Note that a /REsume command allows the re-entry of parameters that were lost due to an unexpected request for operator action:

```
(3) >/EXEC PROGA,1,2
    *SI NOT READY
    >/RESUME 1,2
    *14:07:15
```

Under certain circumstances, a program may not be resumable following suspension by console interrupt. Normal OS action upon a console interrupt is to delay approximately three seconds to allow completion of any current I/O. Then the interrupted location is examined to see if it is within OS' own area of memory. If so, this indicates a malfunction (I/O "hang-up", etc.), and the program is automatically cancelled. Subsequent entry of a /REsume command will be rejected with a "CMND REJECT" message. When this happens, a /STatus command would show that the program has been cancelled.

2.5.13 The /STatus Command

/STatus

This command displays on the command output (CO) device the current program status. This allows the user to query the system regarding the current program, its status and limits.

```
(1) /STATUS
    *TEST1,C1-FB,1400-1700,T,12:14:00
```

As shown in example (1), the system responds with the program's name, its scratchpad and its main memory requirements, its status, and the current time.

```
(2) /STATUS
    *TEST1,C1-FB,1400-1700,S,12:14:00
    *P=1512,A=0000,X=147C,S=0007
```

Under some conditions, such as when a program has suspended itself or was interrupted with the console interrupt, the display will include additional program information. This display is illustrated in example (2) and shows the contents of the program's P, A, and X registers, and the contents of the Status Word.

The status flag (preceding the time) may be one of the following characters:

C	Cancelled
E	Executing SUPV: Request 5, Get Parameter
L	Loaded but not yet executed
N	No program loaded (e.g., following a /JOB command)
S	Suspended
T	Terminated

2.5.14 The /TIme Command

```
          hh:mm:ss
/TIme     hh:mm
          hh
```

This command allows the user to display and/or set the system clock, which is then available to system and user programs and displayed by some system commands. The system clock operates on a 24 hour day and the time is automatically reset at 24:00 hours (midnight).

```
(1) /TIME
    *13:15:36
```

The command may be given without a time parameter, as in example (1). In this case, the user is asking that the current time be displayed.

```
(2) /TIME  3:19:47
    *03:19:47
```

The command may also be given with a time parameter and the system will reset its clock to this value. The system will again display the time to indicate acceptance, as in example (2).

The time parameter fields may each be supplied as one or two digits, indicating 0 to 23 hours, 0 to 59 minutes, and 0 to 59 seconds. Zeroes will be assumed for omitted minutes or seconds.

2.5.15 The /TYpe Command

/TYpe comments

The /TYpe command restores the assignment of the Command Input unit (CI) to its standard physical device, as defined during system generation. It is the equivalent of an explicit assignment command -- /ASSIGN CI=xx -- but provides more flexibility because the user need not know what the standard device for CI happens to be.

2.6 PROCESSOR STOPS WITHIN OS

Certain serious hardware problems will generate processor stops, or "coded halts," within OS. Display of the P Register will show an address too low to fall into the User Area for the generated system. Display of the I Register will show a value of :08 for the high-order byte, and one of the following values in the low-order byte.

:01 -- CI Unit Open Failure

The system attempted to open the physical device designated for Command Input, but was unsuccessful. Check the device in question, and re-load OS.

:02 -- CO Unit Open Failure

The Command Output Unit could not be opened. Refer to Stop :01.

:03 -- Real-Time Clock Inoperable

The Real-Time Clock is either not installed, or not operating correctly. It is not possible to run OS without a clock. This message can occur only when OS is first loaded.

:04 -- Disk Controller Permanent Error

The continued operation of the disk controller is undependable. Notify Computer Automation.

:05 -- Disk Controller Permanent Error

Refer to Stop :04.

:12 -- Memory Parity Error

Notify Computer Automation.

2.7 OS ERROR MESSAGES

When the Operating System detects certain errors, the operator is notified with a short descriptive message. The Command Input (CI) and Command Output (CO) units are temporarily re-assigned to the operator console. If a user program is executing, it is suspended and its hardware registers and status are saved.

After the operator has taken the required corrective action, normal system operation may be continued with the appropriate commands. If a user program was suspended, and it is possible to continue with its execution, a /RESUME command may be used.

The messages described here are issued by the resident Operating System itself. Messages peculiar to each System Utility Program are described in the individual program write-ups. Each message is shown as it appears on the CO unit, with its cause and possible corrective action.

CMND REJECT

CAUSE: OS command statement just entered on CI is not valid. Either the command is not recognizable, or the operands are wrong, or the command cannot be processed in the present context.
ACTION: Correct the statement and re-enter it.

xx NOT READY

CAUSE: A program is attempting to access physical device xx, but the device is not ready for operation. For example, a device is off-line, or a disk, tape, or cassette does not have an OS volume label.
ACTION: Correct the problem and /RESUME.

xx NOT FOUND

CAUSE: Physical device xx or logical device xx is being referenced, but does not exist.
ACTION: If the reference is within a /BATCH or /ASSIGN statement, handle as a CMND REJECT message. If the reference is internal to a program -- for example, an improperly initialized FCB -- /CANCEL the program.

xx UNASSIGNED

CAUSE: Logical unit xx is being referenced, but has not been assigned.
ACTION: Enter an appropriate /ASSIGN, then /RESUME.

flname NOT FOUND

CAUSE: A file-oriented OPEN was attempted for the file with the name shown, but it could not be found. The file name was supplied either in an /ASSIGN command, or in the program's FCB.
ACTION: Either correct the /ASSIGN statement and re-enter it, then /RESUME; or /CANCEL the program.

xx ILLEGAL OPEN

CAUSE: A program has made a request for an illegal operation on physical device xx during an OPEN. (An illegal operation in an IO: request does not cause the message, but takes an error return within the program.)
ACTION: /CANCEL the program.

flname DUPLICATE FILE

CAUSE: A program is attempting to OPEN for WRITE a file with the name shown, but this name is already in use for an existing file on the same device.
ACTION: Either use OS: CPY to delete the existing file, and re-run the program; or change the /ASSIGN command or FCB which supplied the duplicate name.

xx MULT WRITE ERROR

CAUSE: A program is attempting to OPEN multiple output files on the single device xx, which does not allow this technique. The device is probably a tape assigned where a disk was intended by the programmer.
ACTION: /CANCEL the program. Re-run with different assignments, or with the programming technique changed.

xx WRITE PROTECT

CAUSE: xx is a disk, and no space remains for a WRITE operation requested by a program.
ACTION: /CANCEL the program.
CAUSE: xx is a cassette drive, and the address track on the cassette is still write-enabled.
ACTION: Remove Tab A, and /RESUME.
CAUSE: xx is a tape with no write-enable ring, or a disk with the protect light on, or a cassette with no Tab B.
ACTION: Correct the problem, and /RESUME.

xx DATA ERROR

CAUSE: During data transfer, a hardware error occurred on physical device xx which could not be corrected by normal OS retry procedures.
ACTION: Run a device diagnostic program if necessary to correct the problem.

xx HDWR ERROR

CAUSE: During a non-transfer operation, a hardware error occurred on device xx which could not be corrected by normal OS retry procedures.
ACTION: Run a device diagnostic program, if necessary, to correct the problem.

I/O ERR

CAUSE: An unrecoverable I/O error occurred during system operation, probably on SF, CI, or CO. This message will generally follow another error message.
ACTION: Correct the problem, as described for the message accompanying this one.

LOAD ERR

CAUSE: A /LOAD or /EXEC command requested a load of a program from SF which would have exceeded memory, or violated the areas of memory reserved for the system.
ACTION: Check program being requested.

BAD TC

CAUSE: A /LOAD or /EXEC command requested a load of a program from SF, but a Type Code in the program file is not valid for processing by the resident loader.
ACTION: Check the program file. The program probably needs to be processed thru OS:LNK.

POWER-FAIL

CAUSE: Execution has passed thru location :0000, probably because of a Power Fail/Restart sequence. A SPND: call is automatically simulated by the system.
ACTION: /CANCEL the program and re-run it.

2.8 COMMAND EXAMPLES

These examples will illustrate a typical sequence of commands and responses, from system load to job completion. Console messages from OS to the user are underlined.

1. Once the resident components of OS are loaded into memory, and execution begins, the system displays its name and release number, a dummy time, and a dummy date. The operator may enter the actual time and date, or immediately supply the first job.

```
ALPHA/LSI OS (D0)
*00:00:00
*MM/DD/YY
>/TI 9:30:00
*09:30:00
>/DA 9/24/72
*09/24/72
```

2. Request an assembly of a program with source on cards, listing on the line printer, and a file device for intermediate storage.

```
>/JOB ASSEMBLY LOAD AND EXEC
*09:31:00,09/24/72
>/ASSIGN SI=CR,LO=LP,BO=PP,SS=M1
>/EXEC OS:ASM,NCORE
```

3. Request the system to load the program punched above, using the library on unit M0, and execute it.

```
>/ASSIGN BI=PR,LI=M0.LIBRY
>/EXEC OS:LDR,LL,TE
>/BEGIN ,DAILY,3
```

4. After the job has completed, log the total job time for this user.

```
>/NJOB
*09:31:00    09:47:12
>
```

Section 3

SYSTEM UTILITY PROGRAMS

3.1 INTRODUCTION

The Operating System requires a minimum of dedicated core storage. Many useful functions are performed by separate System Utility Programs, which are invoked by a normal /EXEC command as the user requires them.

Most utilities operate in the same way as user programs. They run in the User Area, starting just above the Operating System itself, and extending up to the beginning of the Transient Area in high core. Each program requires certain assignments and parameters. During execution, the entire resources of the system are dedicated to the program.

One System Utility Program, OS:LDR, is executed in the Transient Area to load a user program into the User Area. Its execution is terminated before the user program receives control of the system.

Two special debugging tools, OS:DBG and OS:DMP, are executed in the Transient Area while a user program is resident in the User Area. This allows them to access the user program without overlaying any of the user's own core storage.

3.2 THE OPERATING SYSTEM ASSEMBLERS

Three assemblers are available under OS. They accept a uniform Source Language; the specifications are published separately in OS Assembler Language Reference Manual (96552-00).

MACRO2 generates object code which is intended for an LSI-2, LSI-1, or ALPHA-16. To handle a useful number of symbols and Macro Definitions, more than 16K of memory is ordinarily required.

OS:ASM is a simplified version of MACRO2, intended for OS configurations with a memory size of 16K or less. The most substantial difference between OS:ASM and MACRO2 is the former program's lack of a Macro Facility. Other limitations are described in the Language Reference Manual.

MACRO3 is a variant of MACRO2. The only machine instructions recognized are those meaningful on an LSI-3/05. The generated object code is usable only on an LSI-3/05, usually after processing by the OS Link Editor.

3.2.1 Logical Unit Requirements

SI (Source Input)

Required. Typically a card reader, or a disk file maintained with OS:SFE or OS:EDT. Contains any number of separate Source Programs, each of which must terminate with its own END statement. The Source Input File as a whole must terminate with an End-of-File -- for example, with /* on a card reader.

LO (List Output)

Required unless all listing has been suppressed with an OS parameter. Must be a printer; a magnetic device is not acceptable.

BO (Binary Output)

Required unless all object code output has been suppressed with an OS parameter. Typically a magnetic device, so the file can be turned around to the link editor. If assigned to a paper tape punch, the placement of End-of-File records may be controlled with an OS parameter.

SS (System Scratch)

Required for MACRO2 and MACRO3. Not required by OS:ASM unless OS parameter NC is used. Must be a magnetic device, typically a disk. A normal termination of MACRO2 or MACRO3 will close and delete the file.

SA (Source Alternate)

Not used by OS: ASM. Required for MACRO2 or MACRO3 only if Definition File processing has been requested with an OS parameter. If SA is assigned to the same card reader or paper tape reader as SI, the records for SA must come first, and must have their own End-of-File.

3.2.2 Parameters Available Only for MACRO2 or MACRO3

LL

Load and List the Definition File assigned to SA. The format and device requirements of the file are identical to those of a Source Input File. SA is opened, the entire contents are assembled, and SA is closed and kept. No Binary Output is ever generated. Every definition, symbol, and value established during the SA processing is saved by the assembler, and is considered to be predefined during the processing of the Source Input File.

If this parameter is used at all, it must be the first parameter.

LN

Load with No List. Same effect as LL, except that no listing is produced during SA processing. If this parameter is used at all, it must be the first parameter.

FR

Flag Range Literals. Each source statement which requires an indirect link thru Scratchpad has a Warning Flag "W" on the assembly listing. This parameter is only needed if the LPOOL directive is never used in a Source Program. The presence of at least one LPOOL automatically flags all statements which still need Scratchpad Literals.

3.2.3 Parameters Available Only for OS: ASM

NC

Not Core Only. Forces the assembler to copy each Source Program to the SS file before processing it, even though enough memory may be available to save the whole program without using SS. The result is that SS contains a copy of SI when the assembler runs to normal termination, if these conditions are met:

1. SI is not a named file on a magnetic device.
2. SS is assigned to a named file on a magnetic device.

The point of the NC parameter is that the Source Program need not be put thru a separate run of OS: CPY to create a permanent named file.

P2

Pass 2 Again. This parameter is acceptable only if the assembler has run to normal termination, and a /BEGIN command has been used to restart it. The P2 parameter may be followed by the parameters NL, EL, and NB as needed, on the /BEGIN statement.

The result of repeating Pass 2 is that another copy is produced of the assembly listing, the object code, or both, corresponding to the last Source Program assembled. This is somewhat faster than re-running the entire assembly from SI.

FR

Flag Range Literals. Each source statement which requires an indirect link thru Scratchpad has an Error Flag "A" on the assembly listing. The object code for the statement is still correct.

3.2.4 Parameters Available for All OS Assemblers

NL

No List Output. Prevents the assembler from opening or using LO. The assembler language has a directive called LIST to suppress various types of printed output more selectively.

EL

Error List Only. Nothing is printed except a list of each Source Program's Errors and Warnings.

NB

No Binary Output. Prevents the assembler from opening or using BO.

LI

Library Format on Binary Output. This parameter is superfluous if BO is assigned to a magnetic device. If BO is assigned to a paper tape punch, the assembler ordinarily separates each Object Program from the next with an End-of-File. The LI parameter forces the paper tape into the same Library Format used for BO on a magnetic device -- no EOF between Object Programs, one EOF after the very last Object Program.

3.2.5 Messages on Command Output Unit

name (rr)

CAUSE: Assembly has started. Revision level of the program is rr.
ACTION: None.

INVALID CMND

CAUSE: An incorrect parameter has been supplied.
ACTION: Enter a /RESUME command, supplying all of the correct parameters.

PAUSE

CAUSE: Input ended with an Up-Arrow (↑), rather than an EOF.
ACTION: Ready next segment of the Source Program, and enter /RESUME.

NO END

CAUSE: Software EOF before final END statement.
ACTION: Supply an END statement and another EOF, then /RESUME.
CAUSE: Software EOF after MACRO, but before ENDM was found.
ACTION: Supply an ENDM and another EOF, then /RESUME.

MEM OVERFLOW

CAUSE: The memory shared by symbols, definitions, and literals is exhausted. No more LPOOL entries are made. Out-of-range references generate Scratchpad links and "W" flags. Normal assembly continues.
ACTION: None, unless the programmer chooses to /CANCEL the assembly.

FEED ME

CAUSE: OS:ASM needs the current Source Program for a second pass. The NC parameter was not used. The current Source Program is not available for a second pass on SS, it is too large to have been saved in memory, and the assembler cannot simply re-read it from SI because of the device type involved.
ACTION: Reposition the SI file to the start of the last Source Program read, then /RESUME.

3.3 OS:LDR - THE OPERATING SYSTEM LOADER

The OS Loader Utility (OS:LDR) loads and links together one or more object programs into a single memory-resident program. The resultant program may then be executed, or output using the OS Dump Utility (OS:DMP).

The OS:LDR utility will accept all object programs generated by the BETA assemblers, and all object programs generated by the OS assembler which do not contain references to external labels with an offset.

OS:LDR provides the capability of dynamic peripheral device assignment through the operating system. Depending on the options requested, the loader will require the availability and assignment of the following logical units prior to initiation of the loading process.

1. The SF (system file) unit is required by the system. It is here that the system expects to find the loader program itself.

2. The CI (command input) unit is required for input of loader option requests.

3. The CO (command output) unit is required for printing loader comments and error messages.

NOTE

The SF, CI and CO units were required prior to this step and generally will not require re-assignment at this time.

4. The LO (listing output) unit is required for printing the loader listing. The LO device must not be bulk storage.

5. The BI (binary input) unit is required for input of the main (first) object file.

6. The LI (library input) unit is required for input of library files requested during the load process. This unit is not required if library files are not to be processed.

The OS:LDR utility resides on the System File (SF) and is invoked with the form:

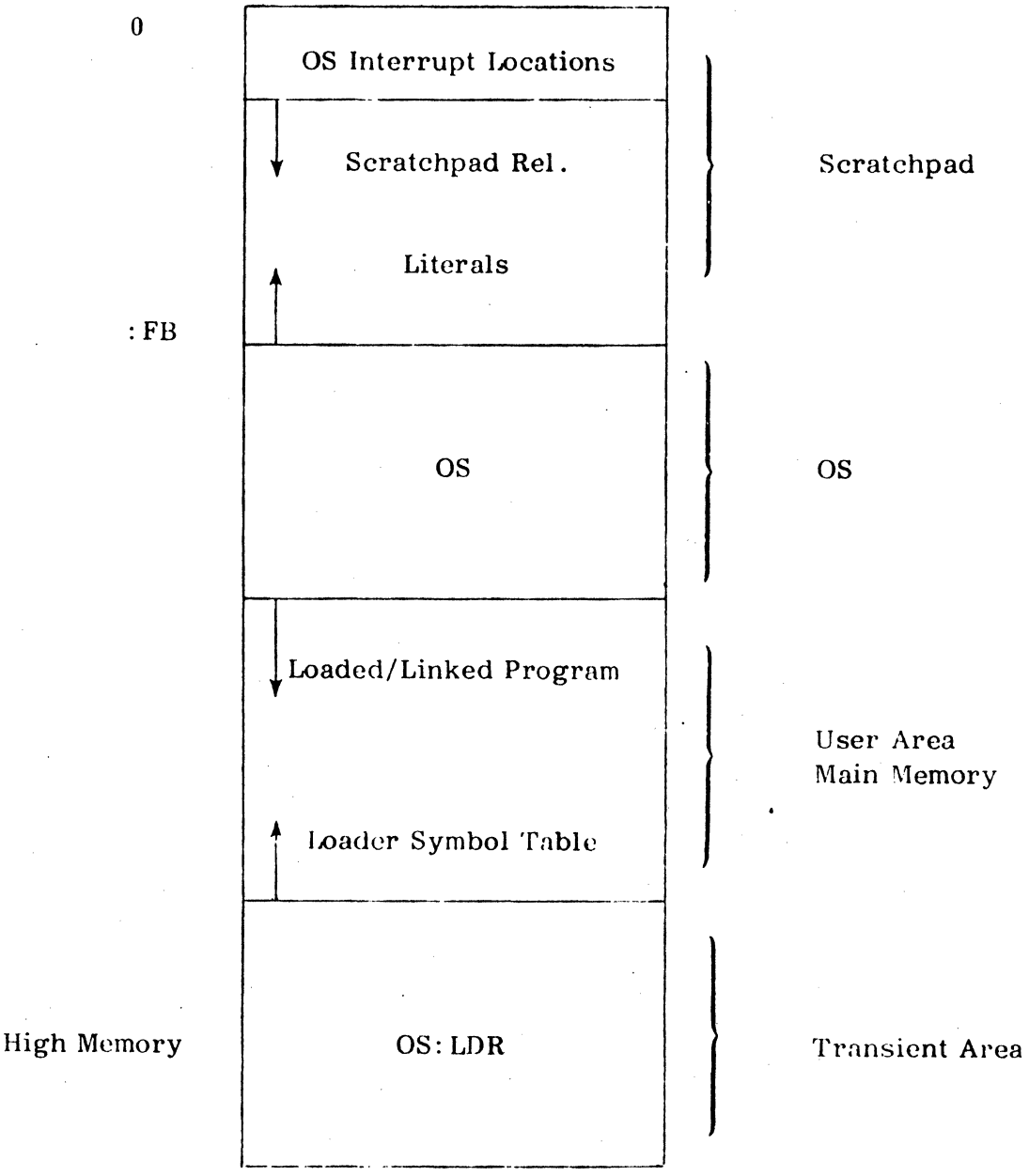
/EX OS:LDR[,option]...

NOTE

The OS:LDR utility is object relocatable and should be added to the System File via the OS:CPY utility. It must be named OS:LDR on the System File directory; no other name is acceptable.

When loaded from the System File, OS:LDR resides in the transient area (high memory) and assumes availability of all memory and scratchpad for load and link processes. Any references to OS subroutine entries (e.g., SUPV:, OPEN:, IO:) are automatically linked to the resident OS. Relocatable input is offset and stored consecutively directly behind OS (main memory and/or scratchpad relocatable). The default store location (biases to REL 0 programs) are:

- Main Memory (MM) — First available location in main memory following OS, continuing toward high memory.
- Scratchpad Literals (SP) — Location : FB, continuing toward : 00.
- Scratchpad Relocatable (SR) — First available location in low scratchpad continuing toward : FB.



OPTIONS

The user may request that the loader perform certain additional or non-standard operations during the loading process. These "options" are entered as parameters on the /EXEC command line or in response to a query from OS:LDR, in the order requested and separated by commas. Each parameter is a single word of two or more characters; the MM, SP and SR options must be followed by an equal (=) sign and a hexadecimal number, with no imbedded blanks.

After OS:LDR is loaded, it processes the command options, as entered, from left to right. The MM, NList, SP, LIst and SR options are processed as they are encountered. The first EXecute, TErminate, ULoad or LLoad option encountered on the /EXEC command line first causes loading of the BI file to commence and then processing of the command. If none of these options are encountered, loading of the BI file begins when the end of the /EXEC command line is encountered.

After a command line has been processed, if a TErminate or EXecute option has not been encountered, OS:LDR will query the operator for more options.

When an invalid option is detected OS:LDR will print the message 'INVALID CMND' and suspend operation. The operator should issue the proper option and any following options on the command line using the /REsume command.

The options available are:

MM = XXXX	Defines the next available main memory bias to be used as hex address XXXX. The command may be entered to load programs in areas other than sequentially directly behind OS.
SP = XX	Defines the next available scratchpad location to be used for literals. The SP option may only be entered before loading of the BI file begins.
SR = XX	Defines the next available relocatable scratchpad location to be used. OS:LDR will assign a value if this is not entered. The default relocatable scratchpad location is acquired from the system (location LOBP in OS ROOT) and is dependent upon the lower base page requirements of the system.
LLoad	Causes library (relative) loading of next LI file. This can be overridden for a single file at a time by use of the ULoad option.
ULoad	Unconditionally loads from the 'LI' (Library Input) file until an end-of-file is encountered, then resumes the library load mode.
NList	Suppresses listing loader information. This command will stay in effect until an LI option is entered, but will be temporarily overridden by a MAp command.

LIst	List loader information on LO (List Output device) whenever an end-of-file is detected on input from BI or LI. This option overrides the NL option.
MAp	Causes the generation of a full load map on LO device.
TErminate	Terminates loading and returns control to the system. The command may be entered any time but will not take effect until after the BI file has been loaded.
EXecute	Executes the loaded program(s) if there was a transfer address; otherwise, the option is rejected. The command may be entered at any time, but will not be processed until after the BI file has been loaded.

MEMORY MAP

A full or partial memory map is generated on the LO device under the following conditions:

1. A list of unresolved primary symbols is generated along with memory usage whenever an end-of-file is encountered from the BI or LI file.

2. A list of defined primary symbols and their definition addresses/values are generated along with memory usage whenever an end-of-file is encountered on BI or LI and there are no unresolved primary references.

3. The MAp option causes the listing of all defined symbols (see 2 above) and undefined symbols.

4. A termination error will cause the generation of a full memory map (see 3 above).

Conditions 1 and 2 can be suppressed by the NList option.

The memory usage list consists of: main memory (MM) used, scratchpad literal memory (SP) used, scratchpad relocatable memory (SR) used and the last execution address (EX) processed (all memory used addresses are inclusive). If the specified memory (MM, SP, SR) or execution address (EX) was not used, the applicable information is not listed. Below is an example of a memory usage map.

MM	1C9B	1CA7	(inclusive main memory 1C9B to 1CA7 used)
SR	0060	006E	(inclusive relocatable scratchpad 60 to 6E used)
SP	00F8	00FB	(inclusive literal area F8 to FB used)
EX	1CA3		(last effective execution address)

TERMINATION

OS:LDR terminates the load-link process when an EXecute or TErminate option is encountered or a termination error occurs. The termination procedure for EXecute and TErminate is to close the LO device, transfer pertinent loader information to OS (see /STATUS command for OS) and transfer control to OS (TErminate option) or the created program (EXecute option).

A termination error is processed as a TErminate command.

TERMINATION ERROR

During the loading process, conditions may occur which will cause the loader to abort. These conditions will cause the program to issue an error message, print a full memory map and terminate.

The termination error message is listed on the CO device in the form '*LDR ER n', where n is the applicable error number listed below.

The list of possible errors and possible solutions includes:

Error Number	Error	Possible Solution
1	Literal scratchpad overflow.	Review use of literals.
2	Invalid loader type code detected.	Use OS:LNK to process all type codes.
3	I/O error	Retry operation
4	A program has attempted to store into OS or a relocatable program in scratchpad has run into OS scratchpad usage.	Review use of memory
5	The OS:LDR (symbol table) and the loading program have collided.	Use OS:LNK to create linked programs.

EXAMPLES OF OS:LDR OPERATION

Example (1) is a typical device assignment to load one program from the BI file and terminate the load process.

```
(1) >/ASSIGN BI=PR,LO=LP
    >/EX OS:LDR,TE
    >
    -
```

Example (2) illustrates: (a) the loading of the main program, (b) the loading of a library file, and (c) the execution of the loaded program.

```
(2) >/ASSIGN  BI=D0.TEST,LI=D0.MATH,LO=LP
    >/EX  OS:LDR,LL,TE
```

Example (3) illustrates the loading of a main program and three library files. The default memory options have also been overridden.

```
(3) >/ASSIGN  BI=D0.MAIN,LI=D0.LIB1
    >/EX OS:LDR,NL,MM=3000,SP=F0,SR=E0,LL
    ?/ASSIGN  LI=D0.LIB2
    ? LL
    ?/ASSIGN  LI=D0.LIB3
    ? LL,MA,TE
    >
```

3.4 OS:LNK - THE OPERATING SYSTEM LINK EDITOR UTILITY

The OS Link Editor Utility (OS:LNK) links together one or more object programs into a single module. The resultant module is generated to an output file, in an object format. This file may then be loaded into memory for execution, using OS, the LAMBDA loader, BLD, or AutoLoad.

The OS:LNK utility will accept as input all object programs generated by the F FORTRAN IV Compiler or any of the ALPHA series assemblers.

OS:LNK can link object programs whose memory requirements differ from those of the host computer, both in total memory size and in specific memory allocations.

OS:LNK allows selective linking of one or more library files, linking only those programs or subprograms requested by previous programs.

OS:LNK normally generates an absolute object program which, when loaded with the OS/EX command, will reside directly behind OS.

OS:LNK will optionally offset the relocatable input by a requested bias (positive or negative) and produce offset absolute or relocatable binary output. Note, however, that absolute input always yields non-offset absolute output.

OS:LNK provides the capability of dynamic peripheral device assignment through the operating system. Depending on the options requested, the link editor will require the availability and assignment of the following logical units prior to initiation of the process.

1. The SF (system file) unit is required by the system. It is here that the system expects to find OS:LNK itself.

2. The CI (command input) unit is required for input of OS:LNK option requests.

3. The CO (command output) unit is required for printing OS:LNK comments and error messages.

NOTE

The SF, CI and CO units were required prior to this step and generally will not require re-assignment at this time.

4. The LO (listing output) unit is the device on which the Link Map and/or link errors are published. Assignment is required unless the NL option is specified.

5. The BI (binary input) file contains the main program (and subprograms) which are to be linked, and must be assigned prior to execution of OS:LNK.

6. The LI (library input) file contains any library subprograms which may be referenced by the programs previously linked from BI or LI. Assignment is required whenever BI is not self-contained and/or more linking is to take place.

7. The BO (binary output) file is required by OS:LNK; it contains the binary module created by the OS:LNK process and must be assigned prior to execution of OS:LNK, unless the NB option is specified.

The OS:LNK utility resides on the system file (SF) and is invoked with a call of the form:

>/EX OS:LNK [,options] ...

OS:LNK assumes operation will be in the host OS computer and assumes available scratchpad and main memory limits based upon this. Any references to OS subroutine entries (i.e., OPEN:, IO:, SUPV:, etc.) are automatically linked to the host OS. Relocatable input is offset and converted to absolute so that it will load directly behind OS.

The user may request that the link editor perform certain additional functions, or that non-standard values be used. These options are of two types. One type may be issued only when invoking OS:LNK and one type may be issued when invoking OS:LNK or in response to a query from OS:LNK. The options are entered as parameters and are separated by commas. Each parameter is a single word of two or more characters of which only the first two characters are required. Imbedded blanks are not allowed.

If the operator enters an invalid option when invoking the OS:LNK utility or in response to a query, the message "INVALID CMND XX" (where XX is the option) will be listed on the CO device and OS:LNK will suspend operation. The operator may reissue the correct option by entering "/RE option (,option)...". Note that the option listed in the message and all that followed it must be re-entered.

The following optional parameters may be appended only when invoking the OS:LNK utility. If any are entered in response to a query they are rejected as invalid commands.

NH
No host

The NH option indicates that the generated program is not to operate under OS in the host computer. Therefore, OS:LNK does not generate linkages to the host OS and assumes available limits of :FD towards :00 for scratchpad literals, a bias of :00 and a high limit of :FD for relocatable scratchpad, main memory limits of :00 to :7FFF, and a bias of :0000 for relocatable programs. If the starting location for scratchpad literals (:FD) or relocation (:00) is not desired, the SP or SR options should be invoked. If a main memory bias of other than :0000 and/or conversion to absolute is desired, the RL or AB options should be invoked.

If the NH option is entered it must come before the options SP, SR, RL, and AB. If NH is entered after these options the message "NH OUT OF ORDER" will be listed on the CO device and OS:LNK will suspend. The operator should re-enter all the options in the correct order by entering "/RE option (,option)...". Any options not re-entered are set to their normal default conditions.

T3
Type LSI-3/05
Program — This parameter indicates that all of the Object Programs being processed are intended for execution only on an LSI-3/05. The T3 parameter is similar to the NH option, and only one of the two may be used during a single link-edit run. The default memory allocations are:

Scratchpad Literals	:00 thru:7E
Scratchpad Indexing	:00 thru :3F
Scratchpad Relocatable	Bias:00, Limit:7E
Main Memory	:00 thru:7FFF
Relocatable	Bias:0000

If any of these allocations are unacceptable, the parameters SP, SR, SX, AB, or RL may be used to override the default values. None of the parameters just mentioned may be entered before the T3 parameter itself, otherwise OS:LNK will suspend with the message "T3 OUT OF ORDER." The operator must then re-enter all of the parameters in the proper order, using the OS command /RESUME. Any parameters not re-entered are set to their default values.

SP = xx
Scratchpad — The link editor will normally start assigning scratchpad literal locations as needed, starting at location :FB for the host OS and :FD for non-host, and progressing towards location zero. The user may override the default location by entering the desired location as a two-digit positive hexadecimal value. The value also then becomes the high limit for scratchpad programs.

NB
No binary — This option suppresses the binary output for the entire link edit process.

The following optional parameters may be appended when invoking the OS:LNK utility or may be entered in response to a query from OS:LNK:

AB = xxxx
Absolute offset — This option causes the generated program to be in absolute binary format, and all relocatable input is offset by the bias specified (where xxxx may be any positive or negative hexadecimal value).

RL = xxxx
Relocatable offset — This option causes the relocatable input to be offset by the bias specified (xxxx) and then output in relocatable format (xxxx may be any positive or negative hexadecimal value).

SR = xx
Scratchpad Relocatable offset — This option causes the scratchpad relocatable input to be offset by the bias specified (where xx is a positive or negative hexadecimal number). This does not alter the low limit for scratchpad literals.

SX = xx Scratchpad Indexing	This option assigns scratchpad indexing locations for LSI-3/05 programs which require indirect indexing pointers. This option must be used in conjunction with T3. Default starting address is :00. The allocated scratchpad locations progress toward location :3F. The user may override the default by entering the desired starting address as a two-digit positive hexadecimal value. This value then becomes the low limit for scratchpad index pointers.
XA = xxxx Absolute transfer Address	This option overrides the normal transfer address and instead uses the absolute transfer address specified (xxxx).
XR = xxxx Relocatable trans- fer address	This option is the same as XA except the specified address is offset by the current relocation bias.
XS = xxxx Relocatable Scratchpad trans- fer address	This is the same as XR except the current scratchpad relocation bias is used. If the transfer address specified is negative (after any offset is applied) then no transfer address is generated.
NL No list	This option suppresses listing of memory map information (e.g., program and common addresses, unresolved references, etc.), except error messages, for the remainder of the link edit process, or until LI is entered.
LI List	This option re-enables the normal listing function, but does not produce a listing at this time (see MA).
MA Memory Map	Generate memory map. Present link information is listed on the LO device. This option allows the user to review link information before link process is completed. The MA option temporarily overrides the NL option.
LL Library link	Begin or resume library link operation with selective linking of currently assigned Library (LI) file. This command is required when Library programs are required to complete the link process, and is generally preceded by an assignment of an LI file.

UL Unconditional link	This option causes all programs located on the currently assigned Library (LI) file to be unconditionally linked. The selective link mode may be resumed after reading of the currently assigned LI file is completed.
TE Terminate	Terminates operation of the OS:LNK utility. Causes output files to be completed and closed, a memory map to be listed and control to be returned to OS. TE must be entered last.

After OS:LNK has been loaded, it will:

1. Process any optional parameters appended to the /EX OS:LNK command up to but not including the first LL, UL, or TE parameter (if they were entered).

2. Unconditionally link all programs located on the Binary Input (BI) file.

3. Any outstanding parameters are then processed. If UL or LL is encountered, programs on the Library Input (LI) device are unconditionally linked (UL) or conditionally linked (LL) as required.

4. At end-of-file on 'BI' or 'LI' if unresolved primary external references exist and no LL, UL or TE parameter is encountered, primary external references are listed on the 'LO' device (unless suppressed by NL) and a request for more parameters is made. 'LI' may then be assigned and/or the next parameter entered.

5. This process continues from step 3 until a TE option is entered.

6. When the TE parameter is encountered all references to Blank Common are resolved and output. Blank Common is defined (if referenced) with the largest size referenced, as the next available higher locations. The Scratchpad Literal Pool is then output and the 'BO' device closed. A memory map is listed on the 'LO' device (unless suppressed). All files are closed and control is returned to OS.

Secondary Reference Processing

If, after each program or subprogram is processed, unresolved secondary references exist, a primary reference is created by OS:LNK to a user-supplied error routine named SRF:ER. If no unresolved secondary references remain, the created reference to SRF:ER (if any) is deleted. Thus SRF:ER may or may not appear on each memory map (under MISSING) as the need for SRF:ER changes.

Since the need for SRF:ER may not be known until the last library program is linked, SRF:ER should be the last program on the last 'LI' file to be processed. SRF:ER will not be needed if no secondary references exist in the programs being linked.

When the TE parameter is encountered, if a program entry point named SRF:ER is defined, all unresolved secondary references are linked to it. All executable references (i.e., LDA X but not DATA X) to SRF:ER are converted to JST SRF:ER instructions.

Error Handling/Recovery

Termination errors cause an error message to be written on the 'LO' and 'CO' devices, a memory map to be generated, and control returned to OS. The messages and their meanings are as follows:

MESSAGE	ERROR CONDITION
*BAD TYPE CODE	Invalid type code detected. The user should restart the link-edit process and/or regenerate the object programs in which the bad type code was detected.
*TABLES FULL	Link edit table overflow. Start over with more memory or do less segmenting of subprograms.
*LINK ERROR n	System error has occurred. The user should inform Computer Automation of the error with as detailed information of the circumstances as possible, including the error number n.
*INVALID CMND XX	Legal parameter limits violated. Reissue correct option by entering /RE option (,option)...

I/O Errors

*I/O ERR	The operating system has detected an irrecoverable I/O error. OS:LNK will terminate operation.
*INPUT CK	An input failure (e.g., high speed reader not ready) has been detected. The operator should ready the input device and enter /RESUME to continue the operation, or cancel (/CA) OS:LNK.

Errors in Input Programs

The following error messages are listed on the 'LO' device as the error occurs. The messages are for information only. OS:LNK continues normal operation after the errors are listed.

*COMMON SIZE CONFLICT, IGNORED, Program Name, Common Name, Size as first defined, size as re-defined

A labeled common with an incompatible size has been detected. The common area is allocated with the size as first defined and if re-defined with a smaller size, no problems should occur. If re-defined with a larger size, references to that common past the end of the allocated area will produce invalid results.

*SCRATCHPAD LITERAL OVERFLOW, IGNORED. Program Name

The Scratchpad literal pool has reached location :00. Additional literals will not be assigned, and references to them will reference location :00.

*SCRATCHPAD USAGE CONFLICT, IGNORED. Program Name, Scratchpad Location

Input data has been encountered that would be placed in a Scratchpad location that is already occupied by a literal or other input data. If a literal has been assigned to this location the literal has priority; otherwise the last data input will be placed in the location. Literals will never be assigned to an occupied location but will instead be assigned to the next lower unused location, so literal assignment will never cause this message to be listed.

*SCRATCHPAD PROGRAM OVERFLOW, IGNORED. Program Name

A program in Scratchpad has passed the high limit of scratchpad (:FB normally, :FD for NH or XX if SP=XX was entered).

*SCRATCHPAD PROGRAM/LITERAL OVERLAP, IGNORED. Program Name, Scratchpad Location

The scratchpad literal pool, working down from the high limit of scratchpad, and a program, working up from the bottom of scratchpad, have passed each other at the location listed.

*MEMORY OVERFLOW, IGNORED. Program Name

The program has gone past the end of memory (:7FFF) and wrapped around to location :0000.

NOT LSI n OBJECT Program Name

An Object Program on BI or LI is not compatible with the machine for which the output Program File is intended. Either an LSI-3/05 Object Program was found, and the T3 parameter is not in control, or T3 is being used, but the Object Program is not acceptable for an LSI-3/05. In either case, the first Primary Entry Name in the Object Program is given, the input is skipped, and processing continues.

Operation Examples

```
(1)  >/AS  BI=PR,BO=DO.OS:ASM
     >/EX  OS:LNK,TE
```

Example (1) illustrates using OS:LNK to put the OS assembler on the system file (DO) instead of doing it with OS:LDR and OS:DMP.

```
(2)  >/AS  BI=PR,LI=PR,BO=PP,LO=LP
     >/EX  OS:LNK,NH,AB=100
     OS:LNK (A0)
     ?LL,TE
     OS:LNK END
```

Example (2) is a normal series of commands to link a relocatable main program (on paper tape) to one or more library subprograms (also on paper tape). The output generated is an absolute program starting at location :100 and is not to operate under OS. When invoked OS:LNK processes the main program (from BI), lists the unresolved external references on the line printer, outputs the (?) character and waits for more commands. At this time, the library tape is readied in the reader and LL,TE is entered causing the process to be completed.

```
(3)  >/AS  BI=D0.MAIN,LI=D1.LIB1,BO=D0.TEST,LO=LP
     >/EX  OS:LNK,NL,LL
     OS:LNK (A0)
     ?/AS  LI=D0.LIB2
     ?LL,LI,TE
     OS:LNK END
```

Example (3) shows the linking of a main program to two library files. Note that LI is reassigned after the main program file (BI) and the first library file are processed. Also note that listing is disabled to suppress printing of unresolved references and then re-enabled to allow printing of the final memory map.

```
(4)  >/AS BO=D0.OBJPRG
     >/EX MACRO3
     >/AS BI=BO,LI=D0,LIB3,BO=PP
     >/EX OS:LNK,T3,AB=100,LL,TE
```

Example (4) illustrates how the output of MACRO3 may be turned around to OS:LNK. The result is a paper tape ready for loading into an LSI-3/05.

Memory Map Key

CREATED FILE
    File Name
        This is the 'BO' file name if any.

MISSING
    Name
        Under "MISSING" are listed all unresolved primary external references (i.e., names on REF or EXTR directives).

PROGRAM
    Name Address
        Under "PROGRAM" are listed the names and addresses of all defined external program entry points (i.e., names on NAM and SNAM directives)

COMMON
    Size Address
        This is the allocated size and starting address of the FORTRAN IV blank common area.

LABELED COMMON
    Name Size Address
        Under "LABELED COMMON" are listed the names, sizes, and starting addresses of all FORTRAN IV labeled common areas.

        All addresses listed above are absolute unless followed by the letter 'R' in which case they are relocatable. All addresses and sizes are in hexadecimal.

```
MEMORY USAGE
     SCRATCHPAD LITERAL  low - high      *EXCEEDS LIMITS BY nnnn
     SCRATCH PAD PROGRAM low - high
     MAIN MEMORY PROGRAM low - high      *EXCEEDS LIMITS
     RELOCATABLE PROGRAM low - high
     EXEC ADDRESS
```

Under "MEMORY USAGE" is a summary of the memory areas to be used by the linked program. The lowest and highest location used in each area is listed. The "EXEC ADDRESS" is the transfer address of the last subprogram encountered that contained a transfer address. If relocatable, the execution address is followed by the letter "R". The "EXCEEDS LIMITS" message is listed if the program or literal pool overflow their allocated areas.

SCRATCHPAD USAGE TABLE

Under "SCRATCHPAD USAGE TABLE" is a picture of the scratchpad usage. The legend at the right of the table identifies the contents of each scratchpad word.

```
PAGE   1     03/29/76  12:49:18  OS:LNK (B1) MEMORY MAP

CREATED FILE  OMEGA

PROGRAM
  CORLM  0002     LINES:  0004     CHARS:  0005     LOWP   0006     TYPSI  000E
  TFF    0013     MXSYM   0015     SCIX    0016     LO     0017     SI     0019
  BO     001A     LOWA    0030     TTLFLG  0041     LOC    0051     REL    0052
  SOURCE 0058     SAVPTR  005C     SD1     00A4     OMEGA2 0100     START  0100
  TTLBUF 0F7F     BIN     109B     CONV1   10A3     CLPA   1114     SAVE   130E
  RETRY  1338     PUNCH   134F     PNCH    137B     FRAME  13A0     BOINIT 13A9
  WEOF   13BD     CNTR1   13CE     BODEV   1414     PKCA   1435     OKCA   145E
  EJECT  1490     PNO     14CC     LIST1   14D1     LIST   14DC     CNTR   1513
  LSF    1515     DOCR    1518     PICTTY  1546     RPTAR  1562     ICH    1583
  TTKIN  15A6     TTKOT   1628     CRDI    168D     ENDALL 16B6     MEMSIZ 16B7

MEMORY USAGE
  SCRATCH-PAD PROGRAM     0000-00F7
  MAIN MEMORY PROGRAM     0100-16CC
  EXEC ADDRESS            0100
SCRATCHPAD USAGE TABLE:
ADDR 0 1 2 3 4 5 6 7 8 9 A B C D E F     LEGEND:
0000 P P P P P P P P P P P P P P P P       A=ABSOLUTE LITERAL
0010 P P P P P P P P P P P P P P P P       B=BYTE RELOCATABLE LITERAL
0020 P P P P P P P P P P P P P P P P       P=ABSOLUTE PROGRAM
0030 . . . . . . . . . . . . . . . .       R=WORD RELOCATABLE LITERAL
0040 . P P P P P P P P P P P P P P P       S=SREL PROGRAM
0050 P P P P P P P P P P P P P P P P       W=WORD RELOCATABLE INDEX POINTER
0060 P P P P P P P P P P P P P P P P       X=ABSOLUTE INDEX POINTER
0070 P P P P P P P P P P P P P P P P       Y=BYTE RELOCATABLE INDEX POINTER
0080 P P P P P P P P P P P P P P P P
0090 P P P P P P P P P P P P P P P P
00A0 P P P P P P P P P P P P P P P P
00B0 P P . . . . . . . . . . . . . .
00C0 . . . . P P P P P P P P P P P P
00D0 P P P P P P P P P P P P P P P P
00E0 P P P P P P P P P P P P P P P P
00F0 P P P P P P P P . . . . . . . .

PROCESSED LSI 2 OBJECT

 NO ERRORS
```

Sample Generated by:

NH, AB=100, SP=FE

3.5 OS:LBL - THE OPERATING SYSTEM FILE LABEL UTILITY

The operating system requires that all file-type devices (magnetic tape, cassette and disk) be labeled prior to use. This involves the creation of "directories" on each individual unit to allow later file processing by name. Do not confuse "labelling" with "formatting" of disk packs and cassettes, which must be done with stand-alone programs before labelling. Refer to Section 4.5.4, Recording Medium Preparation.

OS:LBL operates under the system, and requires user response during the labeling process. The label program requires the availability and assignment of the following logical units prior to program initiation.

1. The SF (system file) unit is required by the system. It is here that the system expects to find the label program itself.

2. The CO (command output) unit is required for printing the label queries and error messages.

3. The CI (command input) unit is required for operator responses.

When executed, the label program will query the user for its variable information.

```
NAME
?
```

The user should respond with a Volume Identification. It must consist of 1 to 6 alphanumeric characters, the first of which must be alphabetic. The ID is terminated with a Carriage Return.

```
TYPE AND UNIT
?
```

The response is the two-character physical device mnemonic (appendix C) of the device which is to be labeled. The response is terminated with a carriage return.

```
DOES XX CONTAIN OS
?
```

If the device to be labeled (XX) contains a copy of the Operating System, the user responds with 'Y' and a carriage return. Otherwise, the user's response is 'N' followed by a carriage return; the next query is suppressed.

```
SAVE OS
?
```

If a system exists on the unit, and is to be saved, the user responds with 'Y' and a carriage return; otherwise 'N' and a carriage return.

If the device to be labeled is a disk, the operator is then asked:

```
NUMBER OF PARTITIONS (1,2,4,8)
?
```

The user now selects the number of partitions into which the disk is to be divided, and enters that value (1, 2, 4 or 8), followed by a carriage return. The number of partitions selected is the limit to the number of files which may be simultaneously open for write operations. For example, the OS assembler requires a minimum of two partitions: one to save the source (SS=D0) and one for binary output (BO=D0).

If OS:LBL determines that the selected number of partitions makes each partition too small, the operator will be notified. This normally occurs only when a partition must be large enough to contain the file directories and OS itself.

```
PARTITIONS TOO SMALL
SAVE OS
?
```

If the operator responds 'N' the space required by OS will be given to partition number 0. If he responds 'Y' the query

```
NUMBER OF PARTITIONS (1,2,4,8)
?
```

will be repeated and the user can request fewer partitions. Each reduction in partitions (e.g., 8 to 4) will double the size of each partition.

NOTE

It is recommended that systems be configured with 4 or 8 partitions, to ensure the capability of multiple files open for writing.

The labeling process then begins. When completed, the user will be offered the option of labeling another device, or termination.

```
LABEL MORE
?
```

If another unit is to be labeled (see note below) the response should be 'Y'; otherwise respond 'N'. Terminate the response with a carriage return.

NOTE

For efficient use of OS, the user should have a labeled file on all storage devices and one back-up file. For example, an Operating System with 3 cassette drives should have 4 labeled cassettes. Note that at least one of the cassettes contains the Operating System as produced by OS:GEN.

3.6 OS:VEW - THE OPERATING SYSTEM FILE VIEW UTILITY

The OS:VEW utility allows the user to display the contents of all file-type devices operatin under the system. At the user's discretion, this "viewing" may be as simple as displayin the volume ID, or as detailed as displaying the entire contents of a given file.

The view utility operates under OS control, and allows dynamic peripheral device assignment through it. The utility requires the availability and assignment of the following logical units prior to program initiation.

1. The SF (system file) unit is required by the system. It is here the system expects to find the view utility itself.

2. The CI (command input) unit is required for input of the utility option requests.

3. The CO (command output) unit is required for printing of utility error messages.

4. The SI (system input) unit is required for reading of the file(s) to be viewed. (The SI unit may also be assigned via a parameter input--see below.)

5. The LO (listing output) unit is required for printing of the contents of the "viewed" file.

(1)
```
>/ASSIGN SI=M1,LO=LP
>/EXEC OS:VEW
```

Example (1) is a typical device assignment sequence prior to execution. The assignment of SF, CI and CO are not shown, since these units were required prior to execution of this step. The example shows that magnetic tape unit 1 is the device to be examined, and that the contents will be printed on the line printer.

Requests are entered as parameters on the /EXEC command line or in response to a console query, separated by commas, and terminated with a carriage return. Functions are performed in the order requested.

1.	V	displays the volume name, creation date and directory limits. This function is automatically performed prior to the first function requested for a given SI device.
2.	N	displays the names of all files on the device.
3.	D	displays the name and directory information for all files on the device.
4.	D.NAME	displays the directory information of the specific file requested. NAME must be a one to six character file name.
5.	F.NAME	displays the directory information and the contents of the specified file. NAME must be included, and must be a one to six character file name.

6. T or TE — terminates the view utility and returns to the system.

7. Dx or Mx or Cx — causes re-assignment of the SI device for all subsequent view functions. "D", "M" and "C" refer to disk, magnetic tape or cassette respectively; "x" represents the unit number. Inclusion of a file name is not permitted.

```
(2) >/ASSIGN SI=M1,LO=LP
    >/EXEC OS:VEW,D,F.TEST1,T
```

Example (2) is a request to view the magnetic tape located on unit 1. OS:VEW will print the volume ID, all directory entries (D), and the contents of a specific file (F.TEST1). The program will then terminate (T).

```
(3) >/EXEC OS:VEW,D0,N,D1,N,TE
```

Example (3) shows how the user might list the names of all active files (N) on two units (D0) and (D1), then terminate (TE).

A "D" function involving a disk file resident in more than one partition will necessitate reading and counting every record and sector in that file. This operation may produce a brief, but noticeable, delay during the printing of the file's directory information.

Certain conditions may cause an error message on CO, and program suspension:

"SI ASSIGN ERROR"	The ASSIGNed SI unit was found to be a non-file-type device, or was ASSIGNed with a file name. The user should re-ASSIGN SI to a file type device without a specific file name, and enter /RESUME, re-specifying the desired parameters.
"INVALID REQUEST"	An illegal parameter was requested. The user should enter /RESUME, specifying the correct parameters.
"XXXX NOT FOUND"	File XXXX was not found on the specified SI device. The user should enter /RESUME, specifying the correct parameters.
"INSUFFICIENT MEMORY"	The record (or block) length of the requested file is too large to fit in available memory. The user may /CANCEL the program or /RESUME, specifying parameters for another file.
"I/O ERROR"	An I/O error status was returned following a request for I/O. The user may enter /RESUME to retry the I/O function.

```
FILE VIEW  MM/DD/YY, 00:04:45


VTOC:                    SYS1DO ,     OS ON UNIT
CREATED                  12/30/74
FILES ALLOCATED:            77
FILES AVAILABLE:           243

SECTORS AVAILABLE:
PARTITION #1          125
PARTITION #2            8
PARTITION #3          717
PARTITION #4         1149

NAME:                              VTOCRD
CREATED:                    12/30/74, 10:11:38
ATTRIBUTES:                   SEQ/BLOCKED
RECORD SIZE (BYTES):                   32
BLOCK SIZE (BYTES):                   512
PARTITION DIMENSIONS:
  PARTITION 1:
    STARTING SECTOR NO.:               41
    RECORDS:                          322
    SECTORS:                           21
  TOTALS:
    RECORDS:                          322
    SECTORS:                           21

NAME:                              OS:VEW
CREATED:                    12/30/74, 10:12:27
ATTRIBUTES:                   SEQ/UNBLOCKED
RECORD SIZE (BYTES):                  510
BLOCK SIZE (BYTES):                     0
PARTITION DIMENSIONS:
  PARTITION 1:
    STARTING SECTOR NO.:               62
    RECORDS:                            9
    SECTORS:                            9
  TOTALS:
    RECORDS:                            9
    SECTORS:                            9

NAME:                              OS:DBG
CREATED:                    12/30/74, 10:12:28
ATTRIBUTES:                   SEQ/UNBLOCKED
RECORD SIZE (BYTES):                  510
BLOCK SIZE (BYTES):                     0
PARTITION DIMENSIONS:
  PARTITION 1:
    STARTING SECTOR NO.:               71
    RECORDS:                           17
    SECTORS:                           17
  TOTALS:
    RECORDS:                           17
```

Sample Generated by: /EX OS:VEW,DO,D

```
FILE VIEW  MM/DD/YY, 00:01:25


VTOC:                    SYS100 ,     OS ON UNIT
CREATED                  12/30/74
FILES ALLOCATED:            77
FILES AVAILABLE:           243

SECTORS AVAILABLE:
PARTITION #1          125
PARTITION #2            8
PARTITION #3          717
PARTITION #4         1149

NAME:                          MACRO2
CREATED:                  01/25/75, 10:34:01
ATTRIBUTES:                 SEQ/UNBLOCKED
RECORD SIZE (BYTES):                  510
BLOCK SIZE (BYTES):                     0
PARTITION DIMENSIONS:
  PARTITION 3:
    STARTING SECTOR NO.:             2426
    RECORDS:                           28
    SECTORS:                           28
  TOTALS:
    RECORDS:                           28
    SECTORS:                           28
RECORD NO.       1
          0100  0004  1D74  0600  6AB2  6F9B  01FB  6C33    .........2......
          E9C6  019A  64B1  FB31  17FB  A41D  DEB2  618A    .F...1....$.↑2..
          F49B  F49B  F4FB  9E32  46E6  01C6  FF84  0231    ............F....
          01F2  5201  5013  D003  108B  E99B  5B00  D099    ......P........P.
          FB00  48B2  589C  01DA  4AFB  8C1D  DEB2  489B    ...2...7....↑2..
          88E2  87B2  479C  03B2  4413  280E  009C  0413    ...2...2........
          D79C  030F  00C6  0199  FA99  F999  F801  1099    W....F..........
          F799  F699  F59B  3AB2  3A9B  3BC6  059A  2FFB    .......2...F....
          711D  DEB2  2E31  01F2  370E  00B3  2913  57DA    ..↑2.......3...Z
          27A3  260F  0092  5C31  02D9  F6F6  0E92  5931    .#.......Y......
          02D9  F5F6  1292  5631  0299  F8F6  1692  5321    .Y..............
          5892  5231  0299  F9F6  1C92  4F31  02DB  16F6    ..............[..
          2092  4C21  6292  4B21  0392  4A31  0599  F999    ................
          F8B2  0E9B  0EF2  1008  0133  FDF6  6504  1DE2    .2..............
          0600  8D28  00FF  FF0D  A028  5FD3  C9D3  C125    ........ ..SISA.
          E932  AAA0  A0E2  0BC6  509C  09B2  349C  07B6    ..*  ..F...2...6
          0E9C  00B6  089C  049C  059C  06FB  2B25  E8F9    ...6.....+%....
          F4B4  0313  C022  03FB  2526  01F9  F4C7  06FA    .4..@.....%.G...
          794B  D865  BEB1  F921  03FB  1C32  A3F9  F4F2    ..X.>1.....#....
          07B6  1F9C  049C  059C  06B1  F821  03FB  1233    .6.......1......
          D5F9  F4C7  06FA  6667  B265  CAF2  11C5  CC06    U..G....?.J..EL.
          FD01  F900  0100  09F8  06FB  EF0A  0B00  0232    ...............2
          B4FF  FFFF  FF03  C02B  6D23  CD24  A124  9BC6    4.....@...M.!..F
          039E  4BFF  091D  DEB6  4DE6  0D9C  05B6  4E9C    ......↑6.....6..
          08B6  510E  0013  289C  0E13  D79C  0D0F  0001    .6........W.....
          109A  36C6  0199  FA01  1099  F399  F299  F199    ...F............
```

Sample Generated by: /EX OS:VEW,D0,F.MACRO2

3.7 OS:CPY - OPERATING SYSTEM FILE COPY UTILITY

The OS:CPY utility program allows the user to copy symbolic (source) and object (binary) files between physical devices, delete files from mass storage devices (cassette, mag tape, or disk), pack disks, merge two or more files, or list a source file on a print device. A "file" is any sequence of records terminated with an End-of-File indicator, as defined in Section 4.5.1.

The OS:CPY utility program operates under the operating system, and allows dynamic peripheral assignment. The utility requires the availability and assignment of the following logical units prior to program initiation.

1. The SF (System File) unit is required to load OS:CPY into memory.
2. The CI (Command Input) unit is required by OS:CPY for input of utility commands.
3. The CO (Command Output) unit is required for listing utility messages.

OS:CPY performs its operation using the logical devices S1 for input and S2 for output. The user should perform an /ASSIGN or /JOB operation, as desired, after the use of OS:CPY to reset the S1 and S2 assignments.

The general command format for OS:CPY is

$$\text{CMND,PID}\left[\begin{array}{l}\text{.NAME}\\ \text{.RSIZE}\\ \text{.NAME.RSIZE}\end{array}\right],\text{POD}\left[\begin{array}{l}\text{.NAME}\\ \text{.RSIZE}\\ \text{.NAME.RSIZE.BSIZE}\\ \text{.NAME.RCOUNT.PARTITION}\end{array}\right]$$

Where:

CMND	One of the valid commands shown below.
,PID	Two-character physical device mnemonic shown in Appendix C, on which the input file is to be read.
,POD	Two-character physical device mnemonic shown in Appendix C, on which the output file is to be created.
.NAME	Name of the file to be read or created.
.RSIZE	Numbers of characters (bytes) per logical record. The default value for RSIZE for input is 80 characters for symbolic and 510 bytes for binary records. The default for output is the input record size.
.BSIZE	Number of characters (bytes) per physical record when blocked. Valid only for symbolic records on bulk devices.
.RCOUNT	Maximum number of records the random file being created may contain. Applicable to disk devices only.

.PARTITION Disk partition in which the random file is to be created. Applicable to disk devices only.

NOTE

In all cases, the input file is read until an end-of-file is encountered, which (except for MB and MS) causes a corresponding end-of-file to be created on the output file.

The commands available with OS: CPY are:

CB,PID [.NAME / .RSIZE / .NAME.RSIZE] ,POD [.NAME / .RSIZE / .NAME.RSIZE]

Copy a binary file from the PID device to the POD device. RSIZE default is 510 characters and blocking is not allowed.

CS,PID [.NAME / .RSIZE / .NAME.RSIZE] ,POD [.NAME / .RSIZE / .NAME.RSIZE / .NAME.RSIZE.BSIZE]

Copy a symbolic file from PID to POD. Input RSIZE default is 80 characters and input BSIZE is not specified; OS will supply it.

OS maintains its internal symbolic files (OS: ASM SS, etc.) blocked 12 logical records per physical record (80/960 bytes) for increased bulk storage efficiency.

CO,PID,POD

Copy all active (non-deleted) files from the PID bulk device to the POD bulk device. The output device must be labeled and any files located on it will be unaltered.

A file on the PID device must not already exist on the POD device by the same name.

DE,PID.NAME

Delete the file specified (.NAME) from the PID bulk device. Deleted files may be physically removed from a disk with the DE and PK commands. A deleted file name may be re-used without errors.

MB,PID [.NAME / .RSIZE / .NAME.RSIZE] ,POD [.NAME / .RSIZE / .NAME.RSIZE]

Merge two or more binary input files into a single binary output file. RSIZE default is 510 bytes. Blocking is not allowed.

When an end-of-file is encountered from the input (PID) file, OS:CPY issues the message "READY NEXT FILE" and suspends operation to allow a new file to be readied. The user continues the operation with the command:

PID [.NAME]

or terminates the operation with an MT command.

MS,PID [.NAME / .RSIZE / .NAME.RSIZE],POD [.NAME / .RSIZE / .NAME.RSIZE / .NAME.RSIZE.BSIZE]

Merge two or more symbolic input files into a single symbolic output file. RSIZE input default is 80 characters and BSIZE is not specified.

OS maintains its internal symbolic files (OS:ASM SS, etc.) blocked 12 logical records per physical record (80/960 bytes) for increased bulk storage efficiency.

When an end-of-file is encountered from the input (PID) file, OS:CPY prints the message "READY NEXT FILE" and suspends operation to allow a new file to be readied. The user continues the operation with the command:

PID [.NAME]

or terminates the operation with an MT command.

PK,PID

Pack PID (disk only) to physically remove all deleted files. This command allows the user to recover all lost disk space due to deleted files. From 5 to 120 seconds may elapse before another command will be accepted, depending upon the amount of undeleted data to be moved.

Do not attempt to pack a disk to which CI is currently assigned, unless the next OS comma is already available from stored parameters without reading CI again:

/EXEC OS:CPY,PK,Dn,/JOB

If an I/O error occurs while packing a disc, the operator can recover most of the files as follows:

1. Execute OS:VEW to determine the remaining file names (N).
2. Execute OS:CPY and copy the files to a temporary device (another disc, paper tape, etc.). If while copying a file an INPUT CK or I/O ERROR takes place, cancel OS:CPY and execute it again with the next file to be copied.
3. Execute OS:LBL and relabel the disc.
4. Copy all the saved files back to the labeled disc.

RB,PID [.NAME / .RSIZE / .NAME.RSIZE],POD.NAME.RCOUNT [.PARTITION]

Create a random binary file on POD (disk only). The maximum number of records (RCOUNT) must be entered, and an optional partition number (1-8) may be entered. If the optional partition number is not entered, OS will assign the first available partition containing sufficient room.

Random files are restricted to one record per disk sector (510 bytes) and blocking is not allowed.

RS,PID [.NAME / .RSIZE / .NAME.RSIZE],POD.NAME.RCOUNT [.PARTITION]

Create a random symbolic file on POD (disk only). The maximum number of records (RCOUNT) must be entered and an optional partition number (1-8) may be entered. If the optional partition number is not entered, OS will assign the first available.

Random files are restricted to one record per disk sector (510 bytes) and blocking is not allowed.

LI,PID [.NAME / .RSIZE / .NAME.RSIZE],POD [.RSIZE]

List a symbolic file from PID to POD (POD should be a listing device). Input RSIZE default is 80 characters and output RSIZE default is 80.

LN,PID [.NAME / .RSIZE / .NAME.RSIZE],POD [.RSIZE]

List with sequential decimal numbers a symbolic file from PID to POD (POD should be a listing device). Input RSIZE default is 80 characters and output RSIZE default is 80 (75 characters from PID, and 5 spaces for line number).

TE

Terminate OS:CPY and return control to OS.

A /JOB or /ASSIGN command should be executed to restore the default S1 and S2 assignments after OS:CPY has terminated as these units are used internally by the utility.

The following examples illustrate usage of OS:CPY under a variety of circumstances:

```
(1) >/EXEC  OS:CPY,CS,CR,M1.TEST.80.960,TE
    >
```

Example (1) illustrates a normal copy operation, from the card reader to mag tape (unit 1) and return of control to OS (TE). Note that the output file is to be named TEST (M1.TEST) and the records are to be written in blocked format 80 byte record into a 960 byte block on M1.

```
(2) >/EXEC  OS:CPY,CS,CR.72,PP
    ?LN,PR,LP
    ?TE
    >
```

Example (2) illustrates cards being copied (first 72 columns) from the card reader to the high speed paper tape punch. The source records are then listed (up to 80 characters) with sequential line numbers.

```
(3) >/EXEC  OS:CPY,DE,D0.TEST
    ?PK,D0,TE
    >
```

Example (3) illustrates the deletion of a file (TEST) from disk drive 0, and then the packing of disk 0. Note that a pack operation need not take place after every delete operation but should take place periodically based on free disk space required.

```
(4) >/EXEC  OS:CPY
    ?CS,M1.TEST,D0.TEST
    ?TE
    >
```

Example (4) illustrates the copying of a file from one mass storage device (M1) to another mass storage device (D0). TEST's attributes (blocked/unblocked and blocking factor) are maintained during the copy process; thus D0.TEST will have the same attributes as M1.TEST.

```
(5) >/EXEC  OS:CPY,MB,PR,M1.LIBRY
    *READY NEXT FILE
    ?PR
    *READY NEXT FILE
    ?M0.MATH
    ?MT,TE
```

Example (5) illustrates the merging of 3 input files (PR,PR,M0.MATH) onto an output file (M1.LIBRY). OS:CPY will suspend operation after each file has been copied, to allow the user to ready the next file. Operation is continued by issuing the physical unit required. The MT command terminates the merge process.

```
(6) >/EXEC  OS:CPY,CS,CR,M1.SRCE
    ?TE
    >
```

Example (6) illustrates the copying of a card file to mag tape (without blocking). It is highly recommended that symbolic files be maintained blocked on bulk storage devices. (see example 1) thus making more efficient use of bulk storage media.

```
(7) >/EXEC  OS:CPY
    ?CO,M0,M1,TE
```

Example (7) illustrates the manner in which the user may request all non-deleted files on a bulk storage device (M0) to be merged onto another bulk storage device (M1).

```
(8) >/EXEC  OS:CPY
    ?RS,CR,D0.RANDOM.500.4
    ?TE
    >
```

Example (8) illustrates the manner in which the user may create a random file of 500 records (sectors) in partition 4.

The file created (RANDOM) will have a fixed length of 500 records. However, it is not necessary to completely fill the file during creation.

During the copy process, conditions may occur which will cause OS:CPY to suspend the current operation. The operator may continue the operation with the following options:

1. Correct hardware problem and enter /RESUME.
2. Enter /RESUME with new OS:CPY command.

The list of possible errors include:

1. INSUFFICIENT MEMORY HHHH — The combined record (and block) lengths required for the operation is greater than available memory. HHHH is the hexadecimal value indicating the additional memory that is needed. The user may correct this condition by requesting smaller records (or blocks) for output files.

2. I/O ERR — The operating system has detected an irrecoverable I/O error. To try the operation again, the user must re-issue the command.

3. INPUT CK — An input failure (e.g. card reader pick fail, high speed reader not ready) has been detected. The operator should ready the input device and enter /RESUME to continue the operation.

4. INVALID CMND — An invalid command, operation, or parameter has been detected in the last command. The operator must re-issue a correct command.

5. ILLEGAL OPERATION — An illegal operation has been detected by the operating system. For example, a request to output binary to line printer, or write to a card reader. The operator must re-issue a correct command to recover from this error condition.

6. END OF MEDIA — In the process of writing to a bulk storage device, OS determined the physical end of the device had been encountered. To correct this condition, the operator should supply another bulk storage media (new cassette cartridge or disk pack) and restart the operation.

3.8 OS:SFE - THE OPERATING SYSTEM SOURCE FILE EDITOR

The OS Source File Editor utility allows for the maintenance (update) of source files supported by the operating system as either hard copy (paper tape) or on bulk storage devices. The resultant files may then be assembled, concorded, saved or retrieved at the user's discretion.

OS:SFE allows corrections (located in the SA unit) to be merged with the initial source (SI unit), creating a resultant source file (SO unit).

The utility will require the availability and assignment of the following logical units prior to operation.

1. The SF (system file) unit is required by the system. It is here the system expects to find the utility itself.

2. The CI (command input) unit is required for input of the option requests.

3. The CO (command output) unit is required for printing of OS:SFE comments and error messages.

NOTE

Units SF, CI and CO will have been used in prior steps and do not generally require assignment at this time.

4. The SI (system ~~output~~ input) unit contains the original source file, against which the corrections will be made. The SI file is not altered during the update process. It may be either a file-type (blocked or unblocked) or non-file-type device.

5. The SA (alternate system) unit contains the commands and source record corrections. These corrections must be presented in the sequence in which they are to be merged with the SI file. This file is not altered during execution of OS:SFE. It may be either a file-type (blocked or unblocked) or non-file-type device.

6. The SO (source output) file is created by the merger of the SA with the SI file, and must not currently exist. The dimensions of the file are determined as follows:

   If SO is a file-type device:

   a. If the SI unit is not a file-type device, the SO file will be blocked 80:960.

   b. If the SI unit is a file-type device, the dimensions of the SI device will be the dimensions of the SO file.

If SO is not a file-type device:

a. If the SI unit is not a file-type device, the SO file will be dimensioned for 80 character records.

b. If the SI unit is a file-type device, the SO file will be dimensioned to the same record length as the SI file.

NOTE

Since in some cases the SO record length will be dimensioned to match the SI record length, OS:SFE may truncate the SA records when they are merged into the SO file, if the SA record length is greater than the SI record length.

7. The LO (list output) unit is required if the user elects to have the SO file listed as it is generated.

```
(1) >/ASSIGN SI=M0.TEST,SA=CR,SO=M1.TEST1,LO=LP
    >/EXEC OS:SFE
```

Example (1) is a representative device assignment sequence prior to program loading. The example shows that the user wishes to update the program located on M0 (TEST) with the corrections located in the card reader and to save the resultant file on M1 (TEST1).

The user may request additional operations to be performed during the edit process. These options are entered as parameters on the /EXEC command line. The parameters may be entered in any order, separated by commas (,). The options available are:

LI	List the SO records on the LO device (72 characters per record, in standard assembler format, with line numbers, unless superseded by any of the other options).
NN	Output to LO, but suppress decimal line numbers.
NF	Suppress formatting of LO output into OS:ASM columns.
n	Restrict LO output to n decimal characters per line, 1 thru 132. If not specified, n = 72 is assumed.

The "NL" option (no listing) used by previous versions of OS:SFE will be accepted but ignored; listing only occurs when explicitly requested.

```
(2) >/ASSIGN SI=M0.TEST,SO=M1.TEST1,SA=CR
    >/EXEC OS:SFE,NN,50
```

Example (2) is similar to the previous example, except that the resultant SO file will be listed without line numbers (NN), and the line will be restricted to 50 characters.

The two valid formats of the source file editor commands are:

```
;AAA  n
;AAA  n,m
```

where the semicolon (;) must be the first character in the record and distinguishes this line as an OS:SFE command. The "AAA" must be the second, third and fourth characters in the record, and one of the allowable commands (see below), and must be followed by at least one space. The "n" and "m" parameters represent the decimal line numbers of the original SI file and must always be greater than the "n" and "m" parameters in the previous command. Where "n" and "m" are both required, "m" must not be less than "n", and no imbedded spaces are permitted between them. "n" and "m" may be any value from 1 to 32767.

The source file editor accepts the following commands from the SA file for the addition, deletion and replacement of source text records:

```
;ADD n
```

Add record(s) - all source records in SA following this command, and delimited by the next command, are inserted in the SO file after line "n" of the SI file. (Note: If ;ADD n,m is input, "m" is ignored.) If the ;ADD command is followed by another command with no intervening source records, the ;ADD command is ignored.

```
;DEL n        ;DEL n,m
```

Delete record(s) - the source record "n" (or records "n" through "m" inclusive) of the SI file will not be copied to the output file. Any source records occurring between a ;DEL command and the next command will be ignored. To delete all the records following source line "n-1" the command ;DEL n,32767 may be given. OS:SFE will eventually suspend itself with the message "SA GREATER THAN SI" as described below, but resuming execution will produce the desired result.

```
;REP n        ;REP n,m
```

Replace record(s) - source record "n" (or records "n" through "m" inclusive) of the SI file will be replaced by the record(s) in SA following this command, and delimited by the next command, in the resultant SO file. If the ;REP command is followed by another command with no intervening source records, the ;REP command is treated as if it were a ;DEL command.

```
;END
```

END indicates the termination of the correction (SA) file. The remainder of the SI file (if any) will be copied to the SO file without modification. Any parameters (n or m) appended to this command will be ignored, and any SA record(s) subsequent to the ;END command will not be processed. Note: An end-of-file encountered on the SA file will serve exactly the same purpose as an ";END" command.

The source and correction files may contain any symbolic source records. Note, however, that each record output to LO, unless formatting is suppressed, will be formatted as a standard OS:ASM source line. The SI line numbers, against which the corrections are made, can be obtained from an assembler (OS:ASM), copy (OS:CPY), concordance (OS:CNC) or previous source file edit (OS:SFE) listing.

Special care must be taken in determining the desired line number for very large files. Listings for OS:ASM, OS:CPY, OS:CNC, and OS:SFE display only the low-order four digits of the line number -- that is, 0000 thru 9999. For lines in the range 10000 thru 32767, the user must decide on the correct high-order digit, and supply a full 5-place line number to OS:SFE.

If a "PAUSE" assembler directive (the first source character of a record is an up-arrow) is encountered, the following action is taken by OS:SFE:

1. If an up-arrow record is read from the SI device:

   a. If the SI device is a file-type device (disk, mag tape or cassette), the record is treated like any other SI record (no pause occurs).

   b. If the SI device is not a file-type device, a SUSPEND call is made with the message "SI PAUSE"; upon RESUME, the record is processed like any other SI record (deleted if it follows a ";DEL" command, passed to SO otherwise).

2. If an up-arrow record is read from the SA device, it is treated as a normal record and no pause occurs.

3. If an up-arrow record is output to the SO device, it is written to the device first, then:

   a. If the SO device is a file-type device, no other action is taken.

   b. If the SO device is not a file-type device,

      (1) A CLOSE call is made for that device (if a paper tape punch, this will cause blank trailer to be punched);

      (2) A /SUSPEND call is made with the message "SO PAUSE";

      (3) Upon /RESUME, an OPEN call is made on the device.

If an up-arrow record is output to the LO device, it is printed unformatted, as if it were a comment card.

During the edit process, error conditions may occur which require operator intervention. In all cases, the program will print an error message on the CO device, and issue a /SUSPEND call to the system. The operator may /RESUME or /CANCEL the edit procedure at this time.

1. INVALID COMMAND - OS:SFE has encountered a semicolon (;) followed by an unrecognized command, or the first record input from SA did not begin with a semicolon, or line number "m" was not greater than "n". The offending line will be printed directly below the error message and OS:SFE will SUSPEND. If the user RESUMEs, any current OS:SFE command will be terminated, and the SA file will be read until a ";" control character is input, whereupon the editing process will continue.

2. SEQUENCE ERROR - An OS:SFE command was input wherein the line number "n" was not greater than that of the previous OS:SFE command. The offending line will be printed directly below the error message and OS:SFE will SUSPEND. If the user RESUMEs, any current OS:SFE command will be terminated, and the SA file will be read until a ";" control character is input, whereupon the editing process will continue.

3. SA GREATER THAN SI - The OS:SFE command was followed by a number "n" (or "m") which was greater than the highest line number in the SI file. The current operation is continued until SI is exhausted, and OS:SFE will SUSPEND. If the user RESUMEs, the editing process terminates as if an ;END command has been encountered

4. I/O ERROR - An OPEN or I/O call was not completed correctly. OS:SFE will SUSPEND. If the user RESUMEs, the I/O call will be retried.

5. ILLEGAL PARAMETER - A parameter following /EXEC OS:SFE was found to be other than "LI", "NL", "NN", "NF" or ($1 \leq$ LO record length ≤ 132). OS:SFE will SUSPEND. If the user RESUMEs, he must include the correct parameters in the RESUME command.

6. MEM OVERFLOW - Insufficient space exists in memory for the I/O buffers and blocking areas required by OS:SFE. OS:SFE will CLOSE each file and SUSPEND. A RESUME command will cause OS:SFE to restart; thus, any options should be re-input behind the RESUME command.

3.9 OS:CNC - OPERATING SYSTEM ASSEMBLER SOURCE STATEMENT CONCORDANCE

The OS:CNC utility program analyzes an assembler language source program. It produces an alphabetized list of symbols, with corresponding definitions and references identified by line number within the source program. The listing, or "concordance," is intended as a supplement to the listing produced by the assembler itself.

The OS:CNC utility is executed under the control of the Operating System. Availability and assignment of the following logical units will be required prior to program initiation:

1. The SF (System File) unit is required to load OS:CNC into memory.

2. The CI (Command Input) unit is required by OS:CNC for input of utility commands.

3. The CO (Command Output) unit is required for printing comments and error messages.

4. The SI (Source Input) unit is required to supply the source program to be analyzed.

5. The LO (List Output) unit is required for printing the concordance. LO must not be assigned to a file-type device.

(1)
```
>/ASSIGN  LO=LP,SI=D1.PROG45
>/EXEC  OS:CNC
```

Example (1) shows a typical command sequence prior to execution. The concordance will be written to the line printer, and the statements are a specific disk file -- probably the output of a previous execution of OS:CPY or OS:SFE.

The default operation of OS:CNC is the analysis of a single assembler source program, terminating with the END statement. Certain optional capabilities may be requested by parameters on the /EXEC statement, as with other OS utilities. The parameters may be entered in any order, separated with commas. Only the first two characters of each parameter are required.

BATCH

Analyze the next source program on the SI unit, and generate a concordance for it. When an END statement is reached, do not terminate execution of SC:CNC. Instead, continue with a new program and a new concordance, with its own page and line numbers. Repeat this process for each source program on SI until an actual end-of-file is reached.

LIST

Before generating a concordance, list every line of the source program being analyzed.

A8

The source program to be analyzed is intended for the ALPHA-8 instruction set only.

Concordance Flags and Messages

The following flags may appear to the left of a symbol definition:

M	Multiple definitions of this symbol were detected
N	NAM directive used this symbol
R	REF directive used this symbol
U	This symbol was undefined and not external
X	EXTR directive used this symbol

For each label in the Source Input file, OS:CNC will list the line number where the label is defined, followed by the label itself and the line number(s) of every other source statement that references the label. Some of the referencing source statements will access the label's memory location without modifying it, other referencing source statements will access the label's memory location and perhaps modify it. For example, LDA TAG will load the A Register from TAG, but will not modify it. Conversely, STA TAG will modify TAG during the store operation, but STA *TAG will not (because the reference is indirect). To aid the programmer in debugging, the line number of each source statement which can (but may or may not) modify the label's memory location is flagged with an asterisk.

An up-arrow (↑) appearing in column 1 of a source statement will cause the same action in OS:CNC as in OS:ASM -- the program will pause until the operator reloads the SI unit and enters the command /RESUME.

If a single source program contains more symbols that OS:CNC can table in available memory, the message "MEMORY FULL" is printed, a partial concordance is generated (showing all symbols defined and referenced to that point), and processing continues. Any partial concordances generated will cumulatively analyze the entire program.

3.10 OS:DBG - THE OPERATING SYSTEM DEBUG UTILITY

3.10.1 Introduction

The OS:DBG Utility is functionally similar to the ALPHA-16 DEBUG (program 96004). OS:DBG resides in the System Transient Area of memory above the User Area. Depending on the options requested, OS:DBG may require the assignment of the following logical units prior to the initiation of the debug process:

1. SF (system file) is required by the system; it is where the system expects to find OS:DBG.

2. CI (command input) is required for OS:DBG command input.

3. CO (command output) is required for printing OS:DBG messages.

4. LO (listing output) is needed for use of the "LIST" command of OS:DBG, and may not be a file-type device.

NOTE

> The SF, CI and CO units have been assigned prior to loading OS:DBG and generally need not be reassigned at this time. Note, however, that since OS:DBG is a conversational utility, CI and CO should, for convenience, be assigned to the same device, nominally the console teletype.

The user program and OS:DBG are both in core during a debugging session. OS:DBG alters, executes, and monitors the user program, working interactively with the programmer at the console. First the user program is loaded, but not begun, then OS:DBG is loaded and entered:

```
(1) >/ASSIGN  LO=LP,BI=PR
    >/EXEC  OS:LDR
    >/EXEC  OS:DBG
    OS:DBG   7BF7
    ?
```

In example (1) the binary input device for the loader was the high speed paper tape reader. The line printer was selected for the loader listing and use of the OS:DBG LIST command. The user program was loaded by OS:LDR, and OS:DBG was then loaded into the system transient area and entered.

3.10.2 Communication with the Program

When first entered, OS:DBG will display its name and re-start location. It will then display a question mark to request a command from the operator. Each command, and each response to a request for a parameter value, will cause some specific action to occur immediately; OS:DBG will then request another command.

A command line consists of a unique single-character function identifier, followed by one or more parameters separated by spaces or commas and terminated by a carriage return. For the convenience of users accustomed to the stand-alone DEBUG program, which expects a period at the end of each command, OS:DBG will also accept the sequence period/carriage return as a command termination. In either case, any number of spaces may immediately precede the carriage return character. If an illegal command is entered, OS:DBG will reject the command with the message "ER" and request a new command.

3.10.2.1 Address and Data Parameters

Any address or data parameter value may be entered as an unsigned, positive, or negative haxadecimal value. Leading zeroes are optional.

```
7FFF   0E08   +50   -4000   0
```

Sixteen offset constants, termed "relocation registers," and described later, are available. In any context where a hex number would be valid, a relocation register reference (the letter "R" followed by a hex digit from 0 to F) is also valid. The reference may be preceded with another hex number as an additional value.

```
R4   600R3   -4RA   -2222R7R4R3
```

A parameter may be entered as the sum or difference of exactly two terms, for convenience.

```
5555+4   -345-2C   4000R7-C700   R4R4+033
```

OS:DBG will recognize that an address parameter -- but not a data value parameter -- with a high-order "1" bit is intended to be used indirectly.

```
A004   2004+8000   8000R6   488R7+8000
```

An incorrect value may be replaced by entering a slash followed by the proper value.

```
1111   2222   3333   5555/4444   5555   6666
```

3.10.2.2 O (Status Word) Register

A location in OS:DBG, accessible to the user and referred to as the O Register, simulates the contents and functions of the hardware Status Word. For the user's convenience, the Status Word is charted here.

BIT	SETTING	MEANING
15--8	0	Unused
7--4		Contents of ALPHA-LSI Console Sense Register
3	1/0	Set/Reset of ALPHA-LSI Sense Switch
2	1/0	Enable/Disable Interrupts
1	1/0	Byte/Word Mode
0	1/0	Set/Reset of Overflow Indicator

On an ALPHA-16, bits 7--4 and bit 3 are unused. OS:DBG will not alter the actual setting of bit 2, regardless of the user's manipulation of the O register.

3.10.2.3 OS:DBG Commands

Each command line starts with a specific command, entered as a single character. Immediately following the command is a set of parameters. Each parameter is separated from the next by exactly one space or one comma, as appropriate for the particular command. The line is terminated with a carriage return, optionally preceded by any number of spaces.

Certain conventions are adopted in this program description to simulate console I/O:

Program Description	Console I/O
CR	Carriage Return (shown in certain examples for clarity; every input line must terminate with a carriage return)
small letters	Hexadecimal Parameter
underline	Output from OS:DBG
(hhhh)	Display of current contents of location hhhh
....	Display similar in meaning to a previous display

A detailed description of each command, with appropriate examples, follows. A summary chart suitable for use at the console is included as section 3.10.4.

Display/Change Relocation Registers. External subroutines are normally assembled at relative location : 0000. When loaded by OS: LDR they are offset, creating a relocation bias which must be added to the location values on the subroutine listing in order to determine the true location in memory of the subroutine. In order to alleviate this problem, sixteen relocation registers are provided (designated R0 thru RF), which may be set to the relocation bias of up to sixteen subroutines. When parameters are suffixed by a relocation register ID (R0 thru RF), the designated bias is added to the value of the parameter. Hence, in order to refer to location 12C of a subroutine loaded at CD8 the user may set R1 to CD8 (the relocation bias), and enter 12CR1 as the parameter. The R command allows the user to set, display, and change the values of the relocation registers. OS: DBG will initialize R0 to the starting location of user-available core.

```
?Rnaaaa
?
```

Set relocation register n to the value aaaa (n = 0 thru F).

```
?Rn
vvvv ?
?
```

Display the value (vvvv) of relocation register n.

```
?Rn
vvvv ?aaaa
?
```

Display the value or relocation register n and then change it to the value aaaa.

Copy Memory. The C function copies one area of memory to another.

```
?Caaaa bbbb cccc
```

Copy locations aaaa thru bbbb (inclusive) to locations starting at cccc. Location aaaa must be lower than bbbb, and location cccc must be outside the area being moved.

Fill Memory. The F function fills a given area of memory with a specified constant, enabling the user to initialize tables and buffer areas.

```
?Faaaa bbbb vvvv
?
```

Store the value vvvv in memory locations aaaa through bbbb inclusive.

Modify Memory. The M function allows the user to enter alterations into memory consecutively starting at a given address.

```
?Maaaa.vvvv
?
```

Store the value vvvv at location aaaa.

```
?Maaaa.vvvv wwww yyyy zzzz
?
```

Store the values vvvv through zzzz in memory consecutively, starting at location aaaa.

NOTE

If the beginning address is not protected, but protected memory is entered later in the operation, the modification will be effective up to, but not including, the protected area.

Inspect/Change Memory. The I function is used to inspect specific memory locations and to make changes to their contents. The locations may be stepped thru quite conveniently, either forwards or backwards. The user enters a starting address, and OS:DBG responds with that same address and its current contents:

```
?Iaaaa
aaaa (aaaa) ?
```

At this point, OS:DBG is waiting for the user to enter a special line terminator, optionally preceded by a new value for the contents of location aaaa. The valid terminators and their meanings are:

;CR	Continue inspection at aaaa+1
,CR	Continue inspection at aaaa-1
CR	Terminate inspection function and accept a new command

The I function will not accept a step which attempts to wrap around the low or high limit of memory. The user must enter a new I command with a new starting address.

The stand-alone DEBUG program accepted a space/CR sequence to indicate a step forward to aaaa-1. This sequence is not accepted by OS:DBG, which ignores spaces immediately preceding a carriage return.

?I3000		Inspect location 3000.
3000	0000?;CR	Do not alter contents of 3000. Inspect 3000+1.
3001	1111?BBBB,CR	Alter contents of 3001 to BBBB. Inspect 3001-1.
3000	0000?,CR	Do not alter contents of 3000. Inspect 3000-1.
2FFF	DDDD?EEEE,CR	Alter contents of 2FFF to EEEE. Inspect 2FFF-1.
2FFE	4444?0CR	Alter contents of 2FFE to 0. Terminate inspection.
?		Ready for a new command.

If the location to be inspected is supplied with a high-order "1" bit (for example, as Iaaaa+8000), multi-level indirect address pointers will be followed down to a direct address word. Only the final directly addressed location and its contents will be displayed. Stepping the same inspection with ;CR or ,CR will then increment or decrement from the final direct address.

Display/Change Pseudo Registers. In debugging the user's program it is often necessary to preset the hardware registers with initial values; upon reaching a breakpoint it may be necessary to inspect and alter the registers. Three pseudo registers, A, X and O (corresponding to the Accumulator, Index, and Status Word registers), are provided. An exit from OS:DBG to the user program causes the hardware registers to be loaded from the pseudo registers; a return to OS:DBG causes the hardware registers to be stored in the pseudo registers.

```
?Aaaaa
?Xxxxx
?O0
?
```

Set the pseudo Accumulator to the value aaaa. Set the pseudo Index to the value xxxx. Set the pseudo Status Word to the value 0.

```
?A
aaaa ?CR
?X
xxxx ?CR
?O
oooo ?CR
?
```

Display the contents of pseudo registers A, X and O.

```
?A
aaaa ?vvvvCR
?
```

Display the contents of pseudo register A. Change it to the value vvvv.

Search Memory/Search with Mask. The S function will search a given range of memory locations, listing the addresses and contents which match a specified bit or word pattern.

```
?Saaaa bbbb vvvv mmmm
cccc (cccc)
dddd (dddd)
eeee (eeee)
?
```

Search memory from location aaaa through location bbbb for the value vvvv and compare only those bits which have a corresponding "1" bit in the mask word mmmm. In this example, the desired value was found at cccc, dddd, and eeee. Omitting the mask is equivalent to specifying FFFF, resulting in a word-level search:

```
?Saaaa bbbb vvvv
cccc vvvv
dddd vvvv
?
```

Search memory from location aaaa through bbbb inclusive for value vvvv and list the location(s) where it is found. In this example the value vvvv was found in locations cccc and dddd.

Print Memory. The P function allows the user to print an area of memory on the CO unit. Up to eight locations together with the address of the first location are printed on each line.

```
?Paaaa bbbb
aaaa (aaaa) (aaaa+1)....(aaaa+7)
aaaa+8 (aaaa+8) (aaaa+9)....(bbbb-1) (bbbb)
?
```

Print the contents of memory locations aaaa through bbbb inclusive.

List Memory. The L function is identical to the P function just described, except that the output is directed to the LO device, rather than CO.

Set Breakpoint and Transfer Control. The B function allows the user to establish one or two breakpoints, then transfer control to his program. When his program reaches either breakpoint OS:DBG will remove both breakpoints, and print the address of the breakpoint and the contents of the registers as of the last instruction executed before the breakpoint. As larger program modules are tested and corrected, breakpoints may be set at longer intervals, until the entire program is debugged.

```
?Bvvvv cccc,dddd
cccc aaaa xxxx oooo
?
```

Set a breakpoint at cccc and dddd. Save the user instructions and store a jump to OS:DBG at these locations. Set the hardware registers from the corresponding pseudo registers and transfer control to the user's program at location vvvv. When either breakpoint is reached, save the hardware registers in the pseudo registers, remove both breakpoints, print the address of the breakpoint reached and values of the pseudo registers A, X and O. In the case illustrated above the breakpoint at location cccc was reached.

```
?Bbbbb,cccc
```

Set two new breakpoints. Transfer control to the user program at the location of the last breakpoint reached.

```
?Bbbbb cccc
cccc aaaa oooo
?
```

Set one breakpoint at location cccc. Transfer control to the user program at location bbbb.

If an error is made in a "B" command, OS:DBG will respond by simulating the occurrence of a breakpoint. This will remind the user that all breakpoints have been removed, and that no new ones have yet been established.

OS:DBG uses absolute location :FF in scratchpad for breakpoint processing. Programs which use high scratchpad should provide for this.

Transfer Control. The J function allows the user to preset the hardware registers from the pseudo registers and transfer control to his program without breakpoints.

?Jaaaa

Set the hardware registers from the pseudo registers and jump to location aaaa.

Enable Modification of Protected Memory. During the normal operation of OS:DBG only the user's area of memory is available for modification. The OS area (including the Executive, IOCS and File Manager) and the transient area in high core (including OS:DBG itself) is protected against modification by OS:DBG commands. For example, the sequence

?Iaaaa
aaaa (aaaa) ?

where aaaa is a location outside the user area, will cause OS:DBG to display the contents of aaaa; but if the next input is a modification of aaaa, the message "ER" will appear, and aaaa will remain unchanged.

This "memory protect" feature may be overridden by the E (Enable Modification) command. Once modification has been enabled, it will remain so until a T (terminate) command is used, or OS:DBG is re-executed. The E command may be input at any time while in OS:DBG.

Trap Function. The OS console interrupt feature (Section 2.3) may be altered so that depressing the switch will cause a return to OS:DBG rather than to the OS executive. This is done by entering, at any time while in OS:DBG, a D command. A terminate (T) command, or re-execution of OS:DBG, will reset the feature to trap to the OS executive.

If used, this altered trap condition must be reset by a terminate (T) command before returning to the OS Executive. Otherwise, the interrupt location will continue to point to the OS:DBG entry point.

Terminate OS:DBG. The T function allows the user to exit from OS:DBG and return to OS. This command will cause the console interrupt feature to be reset to trap to the OS executive if the (D) command has been issued, and will also reset the Modify Protected Memory (E) feature if it has been issued.

3.10.2.4 Error Handling

The message "ER" is output when the following errors occur:

1. An invalid command is entered.

2. An address or data parameter contains a non-hexadecimal character (other than +, -, or R).

3. Parameters are entered out of sequence. For example, a Fill, List, Copy, Print, or Search command is requested, and the second parameter is less than the first.

4. A command is entered which will modify a protected area of memory, and the E command has not been previously entered.

5. An illegal separator appears between parameters, where a space, comma, period, etc. is required.

6. An operation is requested on non-existent memory.

3.10.3 Handling Terminations

Terminations are handled as follows:

1. When debugging a program using OS:DBG breakpoints, a TERM: call to the executive should always be made from OS:DBG rather than from the user program, if possible.

2. If termination must be made from the user's program and it is desired to continue debugging the same program, enter a /BEGIN command for OS:DBG so it can clear any breakpoint previously set up in the user program.

3. If termination is made from the user program rather than from OS:DBG, and a new user program is loaded for debugging, use /EXEC OS:DBG, rather than /BEGIN.

3.10.4 OS:DBG Command Summary

A	Display pseudo A register.
Av	Set pseudo A register to value v.
Ba	Set breakpoint at location a; resume at previous breakpoint.
Ba,b	Set breakpoints at locations a and b; resume at previous breakpoint.
Bc a	Set breakpoint at location a; resume at location c.
Bc a,b	Set breakpoints at locations a and b; resume at location c.
Ca b c	Copy locations a thru b to c and following.
D	Alter console interrupt to trap to OS:DBG.
E	Enable modification of protected core.
Fa b v	Fill locations a thru b with value v.
Ia	Inspect memory location a.
Ja	Jump to location a.
La b	List contents of locations a thru b on LO device.
Ma.v w x	Modify memory starting at location a.
O	Display pseudo O register.
Ov	Set pseudo O register to value v.
Pa b	Print contents of locations a thru b on CO device.
Rn	Display relocation register Rn.
Rnv	Set relocation register Rn to value v.
Sa b v	Search locations a thru b for value v.
Sa b v m	Search for value v using mask word m.
T	Terminate OS:DBG and return to OS.
X	Display pseudo X register.
Xv	Set pseudo X register to value v.

3.11 OS:DMP - THE OPERATING SYSTEM PROGRAM DUMP UTILITY

The OS dump utility outputs the user's program (or other specified area of core memory) in binary format to the device specified. At a later time the user may then reload his "dumped" program using the /EXecute command or the system loader (OS:LDR).

OS:DMP provides the user with the capability of dynamic peripheral device assignment through the operating system. The program will require the availability and assignment of the following logical units prior to initiation of the dump process.

1. The SF (system file) unit is required by the system. It is here the system expects to find the dump program itself.

2. The CI (command input) unit is required for input of the dump option requests.

3. The CO (command output) unit is required for printing OS:DMP error messages.

NOTE

Units SF, CI and CO will have been used in prior steps and do not generally require re-assignment at this time.

4. The BO (binary output) unit is required for output of the binary program generated by OS:DMP.

(1)
```
>/ASSIGN BO=PP
>/EXEC OS:DMP
```

Example (1) is a typical device assignment sequence prior to program loading. The example shows that the user wishes to dump the current core resident program onto paper tape through the high speed paper tape punch.

Should the user desire to save his program on the system for later use, he may dump it to a file device, as in example (2).

(2)
```
>/ASSIGN BO=M0.TEST4
>/EXEC OS:DMP
```

The dump utility assumes certain standard conditions at each request for program dumping A "normal" dump is one in which the current core resident program, both in scratchpad and main memory, is to be dumped as an absolute binary program. If a "start" address was available when the user's program was loaded, it will become the start address of the dumped program.

The user may request, however, that the dump utility program dump alternate areas of core or an alternate start address. These alternate values are entered as parameters on the /EXEC command line, in any order, and separated by commas. Each parameter is a single word of two characters, followed by an equal (-) sign and one or two hexadecimal numbers. The options available are:

1. SP=XX-YY — Outputs the contents of the scratchpad from location XX to and including location YY.

2. MM=XXXX-YYYY — Outputs the contents of the main memory area from location XXXX to and including location YYYY.

3. ST=XXXX — Includes a start location of XXXX in the dumped program. If no start location is desired, the hexadecimal value "FFFF" should be entered.

4. MD=A — Outputs the program as an absolute binary file. The program dumped will reload at the same location, regardless of core limit variations.

5. MD=R — Outputs the program in a binary relocatable format. The dumped program may be reloaded at any available core location. This mode is not available for programs which require scratchpad loader linkage. Thus the "SP=" and "MD=R" options are initially exclusive.

6. CO — Tells the dump utility that a second dump is desired and causes OS:DMP to request further input from the CI device after the current dump request is completed.

```
(3) >/ASSIGN BO=PP
    >/EXEC OS:DMP,SP=A0-FB,MM=1400-2000,ST=FFFF

(4) >/ASSIGN BO=M0.TEST3
    >/EXEC OS:DMP,ST=FFFF,CO
    ?/ASSIGN BO=PP
    ?CR
```

Example (3) is a dump of core from :A0 thru :FB and :1400 thru :2000, with no start address. Example (4) shows how a core resident program could be dumped to magnetic tape unit 0 and to paper tape. In that example, CR indicates a carriage return.

An invalid option request will cause the line to be rejected and the dump utility program to suspend itself. The user can continue by using the /RESUME command, followed by the corrected options. This is shown in example (5).

```
(5) >/EXEC OS:DMP,SP=A0-120
    *INVALID CMND
    >/RESUME SP=A0-FB
```

Should an I/O error occur while dumping, OS:DMP will suspend itself. The user may then /CANCEL or /RESUME the operation as he requires. See example (6).

```
(6) >/EXEC OS:DMP
    *I/O ERROR
    >/CANCEL
```

3.12 OS:ILD - THE OPERATING SYSTEM INDEPENDENT LOADER

The OS Independent Loader utility will load a user program into memory without attempting to protect the OS system area. A memory-resident bootstrap routine which may be entered by the user program will cause reloading of the Operating System. Thus OS:ILD provides a convenient method of loading and executing a non-OS program from an OS-labelled bulk device.

OS:ILD is unique as an OS utility in that it begins execution under control of OS, then relocates itself to high memory and continues as a free-standing loader, which allows OS to be destroyed, if necessary, during the loading of the user's program.

NOTE

Because of the special nature of DMA data transfer to high memory locations, correct loading from a disk file requires certain hardware revision levels (applicable to ALPHA/LSI-2 Processors only). These revision levels are listed at the end of this section. If the user's system is of a lower level than shown, an unrecoverable error may occur during loading (Halt =:0887, X register =:0001; see Error Handling below).

OS:ILD requires the availability and assignment of the following logical units prior to its execution.

1. The SF (system file) unit is required by the system. The system expects to find OS:ILD here.

2. The CI (command input) unit is required for input of the /EXECUTE command and utility option requests.

3. The CO (command output) unit is required for the publication of utility error messages.

4. The BI (binary input) unit is required for input of the file to be loaded. (The BI unit may also be assigned via parameter input--see below.)

The following limitations apply with respect to the user program that is to be loaded.

1. It must reside on a file-type device (OS-labelled disk, magnetic tape, or cassette). Input from paper tape or other non-bulk device is not supported.

2. It must be a binary or object program containing only the following type codes:

Hex Code	Type
:1	Begin program
:2	END absolute
:4	ORG absolute
:6	DATA absolute
:8	RES and store constant
:18	LSI-3/05 Begin program (only if "T3" parameter requested)

This means that a program containing external references and/or REL org'ed data must first be passed through OS:LNK in ABS mode (AB=option) before being loaded by OS:ILD.

3. A program containing more than one END type code -- for example, one produced by the Binary Dump program (BDP) -- may not simply be copied to a file-type device using OS: CPY, because OS: ILD will terminate any subsequent load upon encountering the first END type code. Avoidance of this problem is guaranteed if the program is written to the file-type device using OS: LNK, rather than OS: CPY.

Calling Sequence

OS: ILD is invoked by one of the following command sequences:

```
/ASSIGN BI=PID.NAME
/EXECUTE OS:ILD[,NX,T3]
        or
/EXECUTE OS:ILD,PID.NAME[,NX,T3]
```

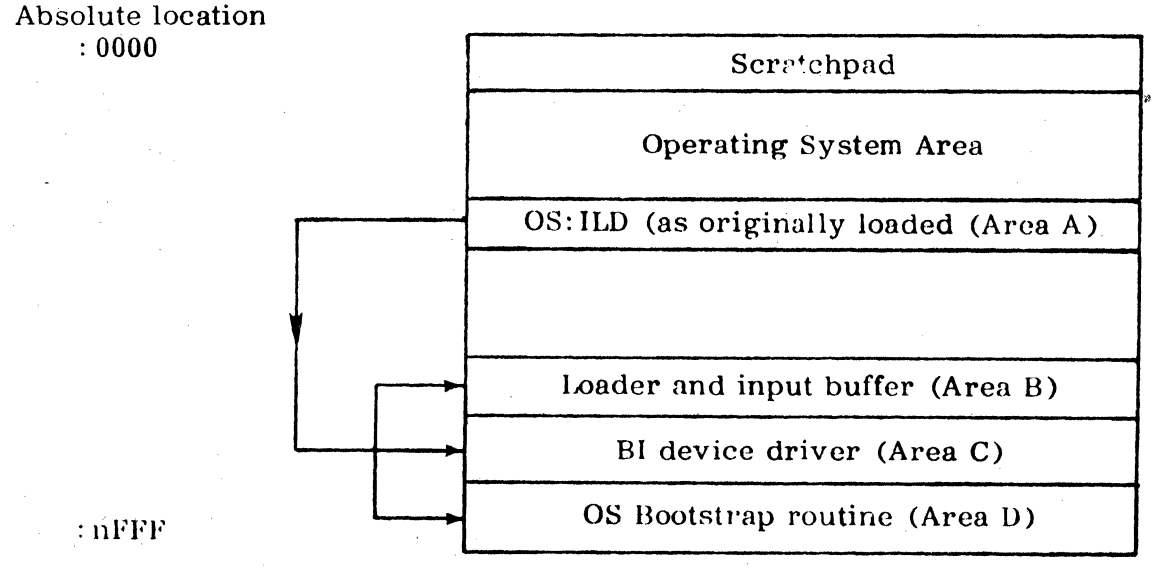
where PID is the OS physical device (Dn, Mn, Fn or Cn) containing the user's program to be loaded (NAME).

NX (no execution) is an optional parameter, signifying that the loaded program is not to be executed; a coded halt will be executed upon completion of loading. If NX is not specified, loading will automatically be followed by execution of the user's program.

"T3" (LSI-3/05 program) is an optional parameter signifying that the program to be loaded is in LSI-3/05 binary format. While such a program would not normally be executed (since OS does not currently run in an LSI-3/05), a non-executing load can be useful in some instances as a tool for verifying loadability, making patches, and/or transferring the loaded memory module into an LSI-3/05 processor.

Program Operation

The /EXECUTE OS: ILD command causes OS: ILD to be loaded by OS into the normal user area above the operating system area. OS: ILD then processes the parameter options, if any, and internally constructs a short bootstrap routine which can reload OS. The Operating System, for the bootstrap's purposes, is assumed to reside on the device which is default-assigned to the SF unit (that device assigned to SF when a /JOB command is executed).



The BI-assigned device is then OPENed, and the first record of the user's file is read, via IOCS, into an input buffer within OS:ILD.

The essential portions of OS:ILD are then re-located into the highest possible memory locations, and control is then transferred to that area for completion of the load process.

At the point when execution is commenced in high memory (after portions of Area A are moved to Areas B, C and D), all interrupts are disabled and coded halt instructions are stored into the power-up and power-down locations. Each record of the user's program is input by means of a short, sense-driven I/O sequence rather than the standard OS input routines. Since at this time the original OS:ILD (Area A) and the Operating System itself are no longer needed, the user has the major portion of memory (from location :0000 up to Area B) available for the loading and execution of this program. The size of Area B is fixed at approximately 404 decimal words in length. Areas C and D vary in size, depending on the BI and SF device assignments:

Area C approximate word length (decimal):

If BI assigned to disk - 83 words

If BI assigned to floppy disk - 91 words

If BI assigned to magnetic tape - 64 words

If BI assigned to cassette - 192 words

Area D approximate word length (decimal):

If SF default - assigned to disk - 19 words

If SF default - assigned to floppy disk - 22 words

If SF default - assigned to magnetic tape - 28 words

If SF default - assigned to cassette - 28 words

Thus in the worst case, where BI and SF are assigned to a cassette unit, all but the last 624 words of memory are available to the user's program.

Recalling OS

Upon completion of loading, the user's program is entered (unless the NX option was specified) with the X register containing the address of the bootstrap routine (Area D). This address is also stored in the last (highest) location of memory. Thus if the user wishes to recall the Operating System after his program has executed, his program should either save the contents of the X register upon entry, or include an address pointer within his program which points to the last memory location. Any subsequent entry into the bootstrap routine will cause immediate loading and execution of OS.

Error Handling

During the preliminary operation of OS:ILD in low memory, while under OS control, the standard OS error handling procedures are in effect. The following error messages will be output to the CO device upon occurrence of the described error:

ILLEGAL PARAMETER

An illegal parameter was input. The only acceptable parameters are the bulk device and user program name, and/or "NX" signifying no execution, and/or "T3" signifying an LSI-3/05 program. Enter a /RESUME command, followed by the correct parameters.

ILLEGAL BI DEVICE

The BI logical unit is not assigned to a disk, magnetic tape or cassette device. /ASSIGN the BI unit to the proper device and /RESUME.

I/O ERROR

An I/O Error has occurred on the first record of the user program, input under control of IOCS. /RESUME to retry the I/O operation.

Once OS:ILD has been relocated to high memory and is no longer under OS control, error indications are output by means of the following coded halt instructions. These halts (except :880 and :881) are also displayed on the LSI console register:

Halt	Description/Recovery
:880	Location :0000 was executed during the load process, indicating a power failure may have occurred. OS must be reloaded through the console, and OS:ILD re-executed.
:881	Location :001C (the power-down interrupt location) was executed during the load process, indicating a power failure may have occurred. OS must be reloaded through the console, and OS:ILD re-executed.
:882	The first type code of the user's file was not a "Begin Program" type code, indicating that the file is not in correct binary format; a source file, for example, will cause this condition. The X register contains the incorrect type code in bits 0-7. Depress RUN to reload OS.
:883	The two input characters following the "Begin Program" type code are not zero, indicating that the file is not in correct binary format. The X register contains the two erroneous characters. Depress RUN to reload OS.
:884	An illegal type code (not :2, :4, :6 or :8) was encountered. The X register contains the illegal type code in bits 0-7. Depress RUN to reload OS.
:885	This halt will occur in the bootstrap routine to reload OS, if the SF default-assigned device was not disk, magnetic tape or cassette upon entry to OS:ILD. OS must be reloaded via the console.

:886 Memory overflow. An absolute load address in the user's program is not less than the start of OS:ILD in Area B, or it is a negative value. The A register contains the start address of Area B, and the X register contains the illegal load address. Depress RUN to reload OS.

:887 Disk I/O error, unrecoverable after ten retries. The X register contains the error status, represented by one or more "1" bits, corresponding to the following statuses:

Bit 0 = 1	Continuously busy
Bit 1 = 1	Disk address ID miscompare
Bit 2 = 1	ID CRC error
Bit 3 = 1	Data CRC error
Bit 4 = 1	End of cylinder error
Bit 5 = 1	Head address error
Bit 6 = 1	Rate error (transmission error)
Bit 7 = 1	Timeout error (incomplete operation)
Bit 8 = 1	Drive not on-line
Bit 9 = 1	Drive unsafe (write check)
Bit 10 = 1	Seek error
Bits 11 - 15	Unused (always zero)

Depress RUN to reload OS.

:888 Magnetic tape I/O error, unrecoverable after ten retries. The X register contains the error status, as follows:

XR = :0000	Tape unit off-line
XR = :0001	Parity error

Depress RUN to reload OS.

:889 Cassette I/O error, unrecoverable after ten retries. The X register contains the error status, represented by one or more "1" bits, corresponding to the following statuses:

Bit 0	Unused (zero)
Bit 1 = 1	Cassette unit off-line
Bit 2	Unused (zero)
Bit 3 = 1	End of tape (runaway)
Bit 4 = 1	Parity error
Bits 5 - 6	Unused (zero)
Bit 7 = 1	Track A address error
Bits 8 - 15	Unused (zero)

:88A Floppy disk I/O error, unrecoverable after ten retries. The X register contains the error status, represented by one or more "1" bits, corresponding to the following statuses:

Bit 0 = 1	Unit not ready
Bit 1 = 1	DMA rate error
Bit 2 = 1	End of cylinder word count not equal to zero
Bit 3 = 1	Attempt to write on write protected unit
Bit 4 = 1	Disk ID miscompare
Bit 5 = 1	ID CRC error
Bit 6 = 1	Data CRC error
Bit 7 = 1	Sync error
Bit 8 = 1	Non-deleted data encountered
Bit 9 = 1	Deleted data encountered
Bit 10 = 1	Operation Complete
Bit 11 = 1	Cylinder zero sensed
Bit 12 - 14	Reserved
Bit 15 = 1	Unit write protected

Depress RUN to reload OS.

:8F0 The user's program was loaded successfully, but not executed because no start address was present on the file. The X register contains the OS bootstrap address. The user must start the program via the console.

:8FF The user's program was loaded successfully, but not executed because the "NX" option was specified. The X register contains the OS bootstrap location. The A register, if positive, contains the user program start address. If negative, no start address was found on the file. (At this point, the A register contents may be altered to reflect any desired start address). Depress RUN to enter the program at the address contained in the A register. The processor conditions upon entry to the user's program are:

Word Mode
Overflow Reset (off)
Interrupts disabled

OS:ILD Required Hardware Revision Levels

73-53500 LSI Motherboard

If level "C" board, revision C6 or higher is required.
If level "B" board, revision B6 or higher is required.
No level "A" board may be used.

73-53506 LSI-2 Processor Full Card

All level "D" boards are acceptable.
If level "C" board, revision C24 or higher is required.
If level "B" board, revision B19 or higher is required.
If level "A" board, revision A24 or higher is required.

73-53507 LSI-2 Processor Half Card

If level "B" board, revision B11 or higher is required.
If level "A" board, revision A9 or higher is required.

73-53531 LSI Disk Controller

If level "B" board, revision B7 or higher is required.
If level "A" board, revision A6 or higher is required.

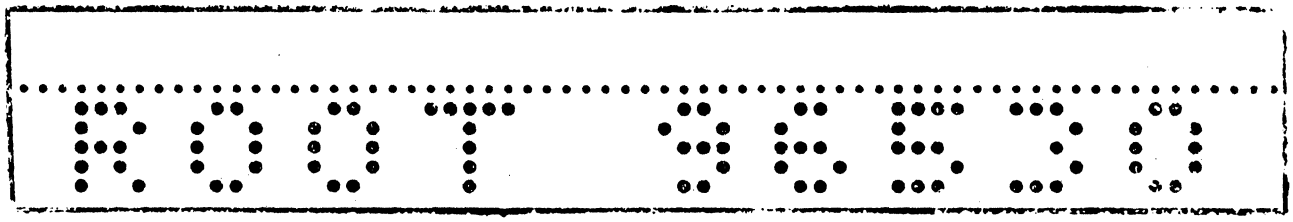
73-53566 LSI Floppy Disk Controller

If level "D" board, revision D2 or higher is required.
If level "C" board, revision C5 or higher is required.
If level "B" board, revision B5 or higher is required.

3.13 OS:HDR - THE OPERATING SYSTEM PAPER TAPE HEADER UTILITY

OS:HDR uses the High Speed Paper Tape Punch to generate Eyeball Headers and other information, based upon literal and symbolic parameters. Ordinarily, OS:HDR is executed just before some other program which punches a data file, such as OS:LNK or OS:CPY, but it may be used to synthesize a complete, usable segment of tape -- for example, a special Bootstrap.

An Eyeball Header is a length of paper tape containing binary configurations, meaningless as data, but arranged so the holes form readable characters. Here is an example:



Any paper tape record read thru CA-supplied software is ignored until a Rubout (:FF) is reached. The 5 by 5 matrix used for each character in an Eyeball Header can never be mistaken for a Rubout, which punches all the channels in one frame of tape.

OS:HDR requires no assignment for its punched output -- it bypasses IOCS and writes directly to Device Address :06, which is assumed to be a High Speed Paper Tape Punch. All commands are entered thru the standard OS parameter mechanism -- appended to the commands /EXEC, /BEGIN, or/RESUME, or supplied in response to a console query.

Commands for OS:HDR fall into 4 categories:

Control -- NL, TE, Logical Unit
Symbolic -- DA, TI, VN, FN, CD, CT
Text Literal
Hex Literal

The rules for punctuating a command are consistent for all categories. A Back Arrow cancels the immediately preceding character, allowing error correction. A Carriage Return terminates the current record on the Command Input device. A Back Arrow immediately before a Carriage Return cancels an entire record. All Line Feed characters are ignored. These rules are standard for OS parameter entry to all programs.

A Comma must be used to separate each command from the next.

If it is necessary to continue across several CI records, OS:HDR requires a Semicolon as a continuation mark, immediately following the last command in a record.

```
>/EXEC OS:HDR,NL,DA;

?'DIAGNOSTIC 210';

?TI,TE
```

CONTROL COMMANDS

NL

No Leader. Ordinarily, OS:HDR runs out 6 inches of empty tape before anything is punched. If this command precedes all non-Control commands, no leader will be punched. If NL is entered after punching has started, it is ignored.

TE

Terminate OS:HDR. This command must be entered eventually, else OS:HDR will continue to demand more records from CI. No empty trailer is punched, but it may be generated explicitly thru a series of blanks in a Text Literal.

Logical Unit

This command consists of any standard 2-character OS Logical Unit Name, such as SI, BO, 06, and so on. Before executing OS:HDR, the Logical Unit is assigned to whatever file is to be referenced when the commands VN, FN, CD, or CT are used.

SYMBOLIC COMMANDS

Each of these 2-character commands will cause the immediate punching, in Eyeball format, of some useful information. The segment of tape punched by each Symbolic command is preceded by a few empty frames, to ensure clarity.

DA

Punch the current System Date, in the OS /DATE format, aa/bb/cc.

TI

Punch the current System Time, in the OS /TIME format, hh:mm:ss.

VN

Punch the Volume Name of the magnetic medium to which the previously specified Logical Unit is assigned.

FN

Punch the File Name to which the Logical Unit is assigned.

CD

Punch the Creation Date of the file to which the Logical Unit is assigned.

CT

Punch the Creation Time of the file to which the logical Unit is assigned.

If the Logical Unit is not a magnetic device, the commands VN, FN, CD, and CT are ignored.

TEXT LITERAL COMMANDS

Each character in a Text Literal is punched in Eyeball format. The Text Literal is delimited by a preceding and a following Single Quote character. If a Single Quote in Eyeball format is needed, it is represented within a Text Literal by the usual convention of using 2 Single Quotes successively.

OS:HDR can translate any printable character in a Text Literal into Eyeball format, including letters, numerals, punctuation marks, and blanks. A Hex Literal can be used to generate any pattern of holes, including readable patterns, which might be useful for special applications.

HEX LITERAL COMMANDS

A Hex Literal supplies characters to be punched in normal binary representation, not in Eyeball format. The paper tape is not spaced out before or after the segment punched for a Hex Literal.

A Hex Literal has a leading Colon, just as in the Assembler Language, and is terminated with a Comma or Semicolon like any other command. The Colon must be followed by an even number of hexadecimal digits. These commands are all valid:

```
:01                     1 Frame
:0000                   2 Frames
:123456ABCDEF003310     9 frames
```

An odd number of hexadecimal digits, even 1 or 3, is not acceptable, because OS:HDR is punching tape frames, not values, and expects 2 digits per frame.

EXAMPLES

1. An existing file on D0, named XYZ, will be punched onto paper tape by OS:CPY. We use OS:HDR to generate an Eyeball Header first, so anyone handling the tape will know what's on it. OS Logical Unit S1 is arbitrarily selected as the connection between the /ASSIGN and the request that the Eyeball Header include the File Name and Creation Date. There is no need to run out any tape after the header, because OS:CPY -- and any other normal program -- will feed some empty tape before the actual data is punched.

```
/JOB
/ASSIGN S1=D0.XYZ
/EX OS:HDR,NL,S1,FN,CD,TE
/EX OS:CPY,CS,D0.XYZ,PP,TE
/NJOB
```

2. A series of Object Programs will be processed thru OS:LNK to create a new Binary Output. We cannot ask OS:HDR to access the File Name and Creation Date, because they don't exist yet, but we can supply the name in a Text Literal, and use the current System Date.

```
/JOB
/ASSIGN BO=PP
/EX OS:HDR,'PCTEST',DA,TE
/EX OS:LNK
```

Messages on Command Output Unit

OS:HDR (nn)
Program execution has started. Version number is nn.

OS:HDR END
Program execution has ended, after the processing of a TE command.

After each of the following messages, OS:HDR will be in a Suspended status. Check the paper tape already punched, and the last command successfully processed. Either enter /RESUME with all the commands still unprocessed, or /CANCEL.

FILE NOT FOUND IN VTOC
The Logical Unit is assigned to a non-existent file.

INVALID CONTINUATION CHARACTER
A command line ends with something other than a Semicolon or a TE command.

INVALID HEX DIGIT
All of the preceding valid frames in the Hex Literal have been punched, but the invalid hex digit pair, and all command information following, must be supplied on the /RESUME.

INVALID LOGICAL UNIT
A command was not a Literal or Symbolic parameter, and does not appear as a valid Logical Unit Name in the OS now running.

INVALID TERMINATOR
A command ends with something other than a Comma or a Semicolon.

I/O ERROR
OS has detected an error condition during I/O processing.

NO CONTINUATION
All commands have been processed, and the last command is not TE.

ODD NUMBER OF HEX DIGITS
The last character of a Hex Literal was not usable. The first command on the /RESUME line should probably be another Hex Literal to specify the intended frame.

UNACCEPTABLE CHARACTER IN TEXT LITERAL
No Eyeball conversion was available for this character. The Text Literal, and all command information following, have been abandoned at this point.

VTOC NOT FOUND
The Logical Unit is assigned to a magnetic device, but no VTOC could be found.

3.14 OS:EDT - OPERATING SYSTEM TEXT EDITOR

SECTION TABLE OF CONTENTS

3.14.1 INTRODUCTION

OS:EDT is a conversational utility for the creation and maintenance of files. Any OS sequential file can be processed thru OS:EDT if it meets these conditions:

Each record contains no more than 128 bytes.

The file is "Symbolic" rather than "Binary" -- it contains printable data, not Object Code or unformatted FORTRAN output.

Some features of the Text Editor are specially provided for files containing Assembler Language or FORTRAN Source Programs, but the Editor can be used to generate documents, test data, OS command files, or other text separated into distinct records.

Contiguous groups of records can be transferred on command between 2 input files, 2 output files, a printer, and the operator's Teletype. Records can also be segregated into 2 dynamically bounded regions of computer memory.

Any record in memory can be located explicitly or by a character-scanning mechanism. Any sequence of characters in any record can be modified, or whole groups of records can be inserted, changed, and deleted by random access.

3.14.2 USING THE TEXT EDITOR -- A SHORT COURSE

The next few pages will present a Short Course on the Text Editor -- a stripped-down version of the facilities used to work with an old file and its listing.

The Text Editor has 3 features which distinguish it from most file-processing programs.

First, records are not read, processed, and written one at a time. Instead, memory is filled with as much input as possible, records are edited in an arbitrary order, and the whole result is written out at once. This cycle is then repeated until the cumulative output file is satisfactory.

Second, a record in memory is identified not by where it is, but by what it looks like -- that is, not by a number, but by any distinctive sequence of characters contained in the record.

Third, once the Editor has been pointed to a record, replacements, deletions, and insertions can be made either to the whole record (and those following it), or to characters within the record.

The Editor's functions are controlled by command lines from the Teletype keyboard -- some letters and symbols, ending with Carriage Returns. A few rules for command lines must be covered first.

A command consists of a single letter, optionally followed by some spaces for easy reading. These commands will appear in the Short Course:

```
N   V   P   D   I   C   L   T
```

When a command needs some additional information -- a parameter -- one of these forms is used after the command letter:

```
$  #  .  'SOMETHING'
```

The first three parameters each identify a record previously read into memory:

```
$   First Record now in memory
#   Last Record now in memory
.   Current Record being pointed at
```

The form 'SOMETHING' represents any sequence of characters, and must be supplied to the Editor with a preceding and a following Single Quote. If a character in the sequence is itself a Single Quote, it must be represented by two successive Single Quotes:

```
'DOG'
'DOG AND CAT'
'AND THAT''S THE TRUTH'
```

A typing error can always be corrected with a Back Arrow, which cancels the preceding character:

```
ABCQ←DE is the same as ABCDE
JKLM98←←NO is the same as JKLMNO
```

A Back Arrow at the end of a line (just before the Carriage Return) cancels the whole line.

Now we're ready to edit an existing file, working from an old listing.

1. Make these assignments:

   SI Source Input -- the existing file
   SO Source Output -- the new file for the edited result
   LO List Output -- the Line Printer

2. Execute the Text Editor. Each new command line will be requested with a Question Mark.

3. Read in a piece of Source Input, and verify the First Record and the Last Record:

   ```
   N  V$  V#
   ```

   If enough memory was available to hold the entire Source Input file, the informative message END OF SI FILE will appear before the First Record is displayed.

4. Point forward to the next record to be modified, point at the start of that record, and verify it:

   ```
   P 'SOMETHING'  P.  V.
   ```

5. To delete a sequence of existing characters, and close up the gap:

   ```
   D 'SEQUENCE'
   ```

   To delete the entire Current Record:

   ```
   D.
   ```

   To delete the Current Record and (for example) 3 more records following it:

   ```
   D.+3
   ```

   Any number from 1 to 32767 can be used.

6. To insert some characters after a sequence of existing characters:

   ```
   I 'OLD SEQUENCE'
   ```

   To insert some records after the Current Record:

   ```
   I.
   ```

   The Text Editor will ask for the new characters or records with a T? type-out. Use Carriage Returns to separate new records. The rules for Single Quotes, and for Back Arrows, still apply. Terminate the new characters or records with a Single Quote and a Carriage Return:

   ```
   T?NEW RECORD 1
   T?AND THAT''S NEW RECORD 2
   T?'
   ```

7. To change a sequence of existing characters:

   C 'OLD SEQUENCE'

   To change the entire Current Record:

   C.

   To change the Current Record and (for example) 3 more records following it:

   C.+3

   Enter the replacement characters or records as described for I commands in Step 6.

8. To continue working forward thru the records in memory, go to Step 4.

9. To list all the records in memory, as a check on the current results:

   L$#

10. If some corrections have been passed by, point at the start of the First Record in memory again:

    P$

    Now go back to Step 4.

11. When all the records in memory are correct, go back to Step 3, which will write the records to the Source Output file and make room for more input.

12. When all processing is completed, returning to Step 3 will result in the message TEXT REGION EMPTY. Close all files and terminate the Editor with this command:

    T

This concludes the Short Course. A detailed formal description of the Text Editor follows.

3.14.3 LOGICAL UNIT REQUIREMENTS

CI (Command Input)

Required for Editor commands. Ordinarily the Teletype keyboard, but any non-magnetic input file can be used.

CO (Command Output)

Required for Error and Warning messages. Ordinarily the Teletype printer, but a line printer can be used. A magnetic device is not acceptable.

LO (List Output)

Required only if the L command is used for listing of text records. Must be a printer; a magnetic device is not acceptable.

SI and SA

Primary and Alternate Inputs. Required only if the commands for each file are used -- R and N for Primary Input from SI, A for Alternate Input from SA. Any input device can be used for either file, or for both. Each file can be blocked or unblocked; the maximum record size allowed is 128 bytes.

SO and S1

Primary and Alternate Outputs. Required only if the commands for each file are used -- W and N for Primary Output to SO, O for Alternate Output to S1. Any output device can be used, including a printer or a punch. The maximum size for an unblocked output record is 128 bytes. Magnetic device output files are automatically blocked in the standard Source File format -- 80 bytes per record, 960 bytes per block.

3.14.4 THE EDITING PROCESS

3.14.4.1 Command Lines

Communication with the Text Editor is thru command lines -- complete logical records on the device assigned to CI. If CI is the Teletype keyboard, each new command line will be requested with a Question Mark on CO, which is ordinarily the Teletype printer. CI can be assigned to any unblocked file; only the first 72 characters of each record are used.

If the command lines are being supplied thru the Teletype keyboard, the usual OS rules for punctuation apply. A Back Arrow cancels the immediately preceding character, allowing error correction. A Carriage Return terminates the current command line. A Back Arrow immediately before a Carriage Return cancels the entire line.

Each command line can contain any number of commands, and typically has 2 or 3 related commands. Each command is a single character, which may or may not be followed by a parameter. Most commands are alphabetic, and no parameter can ever start or end with an alphabetic character. As a result, commands and parameters can be spaced out for easy reading, or squeezed together for faster typing. These examples -- all of which are equivalent -- demonstrate some of the possibilities for a command line:

```
R   D   + 3   P   'ABC'   X   + 4   D

RD+3P'ABC'X+4D

R  D+3  P'ABC'  X+4  D
```

The last line shows the style used in this publication for most examples -- parameters close to their commands, commands separated by one or two blanks. Examples will always have output from the Editor underlined, to distinguish it from command input. In certain cases, the letters "cr" will be used to emphasize that a Carriage Return (or some other way of ending a record on CI) is a significant part of the example. Explanations for each command will always appear to the right of a command line:

```
?R  F'GO'  I  cr     Read some Primary Input.
                     Find the label GO.
                     Insertion will be supplied on next command line.
```

Once the Text Editor has obtained a command line, each command is validated and processed successively. This means that a line with 6 commands -- 2 good, 1 bad, 3 good -- will cause 2 complete Editor actions, followed by a diagnostic message and the cancelling of all the rest of the line.

If an error occurs somewhere beyond the first command on a line, the Editor will log out the already processed commands, with a Question Mark where the line was abandoned. In this example, the non-existent command "Z" is used after some valid commands and parameters:

```
? R D+3 P-1 Z V W cr     Command line entered.
E10 INVALID COMMAND      Diagnostic message.
R D+3 P-1 ?              Commands already processed, and rejection.
?                        Editor is ready for a new command line.
```

Diagnostic messages from the Text Editor are numbered and charted for ready reference at the end of this program description.

3.14.4.2 The Text Region and the Cursor

The highest memory available to the Editor is used for an area called the Text Region. Most Editor commands -- for input, record or character modification, and output -- refer to some point within the Text Region.

Input is not performed on a record-by-record basis. Instead, a large piece of the input is transferred by one command -- enough to fill the Text Region. Then each command to change a record or a character can arbitrarily access any data in the entire Text Region; there is no requirement that a relative order be observed. Output commands can specify any group of records in the Text Region, skipping back and forth as needed. The editing process is repeated on each successive piece of the input until a complete cumulative output file has been created.

Associated with the Text Region is a variable location pointer, called the Cursor. As various commands affect the Text Region, some specific character, somewhere in the Text Region, always has the Cursor "under" it. The Cursor can be at the start of any record:

```
RECORD C        RECORD D        RECORD E
```

The Cursor can be under a character in the middle of a record:

```
RECORD C        RECORD D        RECORD E
```

The Cursor can be attached to a record, but under a position just beyond the last character:

```
RECORD C        RECORD D        RECORD E
```

As the Cursor slides under a record, or jumps from one record to another, the position over it is called the Current Character. Several Editor commands refer to the Current Character, and assume that previous commands have already moved the Cursor to the right place. Similarly, the record in which the Current Character is embedded is called the Current Record, and several commands expect the identity of the Current Record to be established by previous commands.

3.14.4.3 The Save Region and the Command Region

To move a group of records from one point to another within the Text Region, and for other special purposes, the Editor provides a dynamically allocated area called the Save Region. The memory involved is immediately under the Text Region, so the more the Text Region expands, the more the Save Region contracts, and vice versa.

No input, output, or editing can be done on records in the Save Region, which is affected only by the two commands S and U -- that is, Save and Unsave. For further detail, refer to section 3.14.6.3, Using the Save Region.

The fixed part of the Text Editor includes an 72-byte area called the Command Region. A special command -- E -- informs the Editor that the next record on the CI file is to be copied into the Command Region and held for future use. Later, another command -- X -- can use the entire Command Region as if it were a subroutine of the Editor. For further detail, refer to section 3.14.8.1, Command Loops.

3.14.4.4 Reference Points

A Text Region record can be identified in terms of its location relative to a Reference Point. There are three Reference Points, and each has a special character to symbolize it in all command parameters:

$ First Record

. Current Record

Last Record

There are no records before the First Record, and no records after the Last Record. The identities of these records can change at any time, however, as insertions and deletions are applied to either end of the Text Region.

Here are 5 Text Region records, and all the symbolic parameters by which they can be identified. Which Reference Point is used in a command is really a matter of style -- to the Editor, they're all equivalent.

Text Region Data	Relative to First Record	Relative to Current Record	Relative to Last Record
A1A1	$	.-2	#-4
B2B2	$+1	.-1	#-3
C3C3	$+2	.	#-2
D4D4	$+3	.+1	#-1
E5E5	$+4	.+2	#

3.14.4.5 Context Scanning

A Text Region record or character location can be identified by its context. Any distinctive sequence of characters, called a Context String, can be used as the parameter of most Editor commands. The Editor will scan from the current Cursor position, right thru to the end of the Text Region if necessary, until an exact match is found for the Context String.

A Context String is delimited by a preceding and a following Single Quote. Within the string, a Single Quote is represented, as usual, by two successive Single Quotes. The rules for error correction with a Back Arrow still apply. A Context String cannot be spread over several command lines; it is limited to the same line as the command for which it is the parameter.

For a sequence of Text Region characters to match the Context String, they must all appear within a single record. A sequence which is split across record boundaries will never satisfy a context scan.

Suppose the first 5 records in the Text Region look like this:

```
ABC45678
90123ABC
45ABC678
901234AB
C5678901
```

If the Cursor has been positioned at the very start of the Text Region, then a scan for the Context String 'ABC' will be satisfied immediately. If the Cursor is past the first 'A' in the first record, the next match will be the last 3 characters in the second record, and the next after that will be in the middle of the third record. The 'AB' and the 'C' split across the fourth and fifth records cannot match the Context String 'ABC' regardless of the command involved.

If is often convenient to specify that a position within a Context String can match against any character. A Question Mark is used in each "Don't Care" position.

```
ABCDEFG
AB.DE:G
AB DE G
```

For an occurrence of any one of these sequences, the appropriate Context String would be:

```
'AB?DE?G'
```

Context Strings with Question Marks are usually part of Command Loops, as described in section 3.14.8.1.

3.14.4.6 Moving the Cursor

The Position command -- P -- moves the Cursor to any record or character location in the Text Region. The Find Label command -- F -- moves the Cursor to a record with a specified Assembler Language or FORTRAN Label Field.

If the Cursor is already somewhere in the right record, specify a new position with a character count forward or backward:

```
P+3                 Move Cursor forward 3 positions.
P-20                Move Cursor backward 20 positions.
```

The count can be any number from 1 to 32767. The Cursor slides from one record to the next in either direction. The Editor rejects any attempt to cross the low or high boundaries of the Text Region.

If a P command has no parameter at all, the Cursor moves after the end of the Current Record, but does not slide into another record. Backing up one position will then put the Cursor under the last character actually in the record:

```
P                   Move Cursor after end of Current Record.
P P-2               Move Cursor after end, then back 2, so it's under the next-
                    to-last character.
```

There are two ways to jump the Cursor to another record and make it the new Current Record. If it's convenient to count off the new record location relative to a Reference Point, use this technique:

```
P $                 Move Cursor to start of First Record.
P #                 Start of Last Record.
P .                 Start of Current Record.
P $+32              Start of 32nd record after First Record.
P .-3               3rd before Current Record.
P .+5               5th after Current Record.
P #-7               7th before Last Record.
```

Alternatively, let the Editor scan for a context match. The Cursor will be just after the matching characters, and its exact character position in the record can be re-adjusted as needed:

```
P 'LDA REC'   P.    Move Cursor after first match against string.
                    Move Cursor to start of Current Record.
```

If the match is against the very last characters in a record, the Cursor will be after the end of the record, as described for a P command with no parameter.

A context scan starts at the Current Cursor Position, and runs forward to the end of the Text Region. Make sure the Cursor is located where a forward scan will find the match:

```
P $+40 P 'XY' P.+1  Move Cursor to Start of First Record plus 40 more.
                    Move Cursor after first match against string.
                    Move Cursor to 1st record after (new) Current Record.
```

The Find Label command always tries to match against the leftmost positions of a record, and its parameter is limited to 1 thru 6 characters. The search begins with the Current Record, even if the Cursor is beyond the label area. The Cursor is moved to the start of the first matching record:

```
F 'MAIN'          Move Cursor to the start of the first record with 'MAIN'
                  starting in position 1.
```

This command is typically used for an Assembler Language program, but neither the context string nor the text need resemble that language:

```
F '/AS SI'        Find the first record with '/AS SI' in positions 1 thru 6.
F '  '            Find the first record with 2 blanks in positions 1 and 2.
```

For historical reasons, a FORTRAN statement is allowed to have its label -- an unsigned decimal value -- floating anywhere between positions 1 and 5, and to have blanks arbitrarily inserted between the digits of the statement number. Special provision has been made in the Editor for scanning FORTRAN labels -- an F command with a decimal parameter without quote marks will scan for a match according to FORTRAN rules:

```
F 12              Find any of these labels anywhere in positions 1 thru 5:
                      12
                       012
                        1 2
                      etc.
```

3.14.4.7 Checking the Cursor Position

The Q command displays the Current Record up to the Cursor, which is represented on the type-out by a Back Arrow:

```
Q                   Display Cursor.
ABC←                (Current Record up to Cursor)
P+3  Q              Move Cursor forward 3 positions.
                    Display Cursor.
ABCDEF←             (Current Record up to Cursor)
P $   Q             Move Cursor to start of First Record.
                    Display Cursor.
←                   (Current Record up to Cursor)
```

The last display is not very helpful, so the Q command accepts an optional parameter -- a Plus Sign indicates that the entire Current Record must be displayed, with a Back Arrow for the Cursor overlaying the Current Character:

```
Q+                  Display Cursor.
←23456789           (Current Record)
P  P-3  Q+          Move Cursor after end of Current Record.
                    Move Cursor backward 3 positions.
                    Display Cursor.
123456←89           (Current Record)
```

If a Q+ command results in a full record with no Back Arrow, the Cursor must be in the special position between the end of the Current Record and the start of the next record:

```
P '9'  Q+           Move Cursor after first match.
                    Display Cursor.
123456789           (Current Record)
```

3.14.4.8 Supplying New Text in Command Lines

Data can be copied directly from the Command Input device into the Text Region with these commands:

Insert Record Group
Change Record Group
Insert Characters
Change Characters

The function of each command is described elsewhere in this publication. This section deals with the 3 ways of supplying New Text to be used for insertion or change.

New Text always ends with a Single Quote. As will be seen, it may or may not start with a Single Quote, depending on how the New Text is entered. Within a New Text string, a Single Quote is represented, as usual, by two successive Single Quotes. The rules for error correction with a Back Arrow still apply.

If a New Text string will fit on the same command line as the associated I or C command, it can be entered in this immediate format:

First, a Comma
Second, a Single Quote
Third, the actual New Text
Fourth, another Single Quote

Once the terminating Single Quote has been entered, the command is satisfied, and the Editor expects either another command, or a Carriage Return:

I $, '***' V$+2	Insert after the First Record, a new record consisting of 3 asterisks. View the first 3 records now in the Text Region.
C#-1,' END'	Replace the last 2 records in the Text Region with 1 record, an END statement.
C+2, '0'	Replace 3 characters -- the Current Character and 2 more following -- with a single zero.

If it seems more convenient to enter the New Text in a line separate from the command, the line-by-line format is available:

First, a Carriage Return
Second, one or more lines of New Text, each ending with a Carriage Return
Last, a terminating Single Quote and Carriage Return, either attached to the last line of New Text, or on a separate empty line

When the Editor recognizes this format, by finding a Carriage Return where it expected a Comma, each New Text line will be requested with a T? type-out:

? I $ cr	Insert after the First Record in the Text Region.
T?NEWcr	Editor asks for New Text. One record entered.
T?ANOTHERcr	Editor asks for more New Text. Another record entered.
T?'cr	Record entry terminated by Single Quote and Carriage Return.
? C#-1 cr	Replace the last 2 records in the Text Region.
T?XXXcr	One record entered.
T?YYYcr	One record entered.
T?ZZZ'cr	One record entered, and record entry terminated.
? C+5cr	Replace the Current Character and 5 more following.
T?01cr	One segment of replacement characters entered.
T?23cr	Another segment of replacement characters entered.
T?4567'cr	Last segment of replacement characters entered.

In the last case just shown, observe that all three New Text lines are concatenated into a single replacement string, because the command specified a character replacement -- 6 characters were replaced by "01234567" within the Current Record.

The third way to enter New Text is the mixed format. The command supplies the first segment of the New Text in the immediate format, but without a terminating Single Quote before the Carriage Return, then continues in the line-by-line format:

```
C +1, 'ABCDEcr      Replace Current Character and 1 more following.
T?FGHIcr
T?JKLMN'cr
```

The result here is the replacement of 2 characters with 14 characters within the Current Record. An open-ended command like the first line in this example can be useful in a Command Loop, as described in section 3.14.8.1.

3.14.5 RECORD GROUPS

3.14.5.1 Record Group Parameters

A number of Editor commands access a whole group of Text Region records at once:

C	Change Records
D	Delete Records
L	LO Output
O	Sl Output
S	Save Records
V	CO Output
W	SO Output

For these commands, an acceptable parameter has one of these formats:

$	First Record only
$+n	First Record and n more following
.-n	Current Record and n more preceding
.	Current Record only
.+n	Current Record and n more following
#-n	Last Record and n more preceding
#	Last Record only
$.	First Record thru Current Record
.#	Current Record thru Last Record
$#	First Record thru Last Record
	-- that is, the entire Text Region

Here are some examples of Record Groups:

L $+29	List first 30 records.
D #-6	Delete last 7 records.
V.	Display Current Record.
S$# D$#	Copy Text Region to Save Region.
	Clear Text Region.

3.14.5.2 Replacing a Record Group

A Change command -- C with a Record Group parameter -- replaces a series of contiguous records with new records from CI. The new group need not have the same number of records as the old group.

After a Change Record Group command, the Cursor is moved to the start of the final replacement record.

The Editor accepts the replacement records in any of the New Text formats described in section 3.14.4.8. The line-by-line format is used in these examples:

```
C. cr                 Change Current Record only.
T? REPLACEMENT cr     Editor asks for New Text.  One record entered.
T? 'cr                Editor asks for New Text.  Record entry is terminated.
C #-5 cr              Change Last Record and 5 more preceding.
T? XXX cr             One new record.
T? YYY cr             One new record.
T? 'cr                Record entry terminated.
                      6 records have been replaced by 2 records.
Q+                    Check Cursor.
→YY                   (Current Record)
P$ P' END ' C. cr     Position the Cursor at the start of the Text Region.
                      Scan for the next record with a matching string.
                      Change that record.
T? TAG  END  XFR cr   Editor asks for New Text.  One record entered.
T? 'cr                Record entry terminated.
```

3.14.5.3 Deleting a Record Group

A Delete command -- D with a Record Group parameter -- permanently eliminates a series of contiguous records from the Text Region, and compresses the remaining records. The Cursor is moved to the start of the first record after the deletions:

```
D .                 Delete the Current Record only.
D $+1               Delete the First Record and 1 more following.
D $#                Clear the entire Text Region.
D #-3               Delete the last 4 records in the Text Region.
```

In cases like the last example, where no records follow the deleted group, the Cursor is moved to the start of the Last Record.

A Delete command is often guided by a preceding context scan with a P or an F command:

```
P '/TEST/'  D.+1    Move Cursor to next record with matching string.
                    Delete the record and 1 more following.
F '*'  O. D.        Find the next record with an Asterisk in Column 1.
                    Log the record on the S1 file.
                    Delete it from the Text Region.
V $#                View the entire Text Region.
111                 (Editor types out all the records)
333
111
555
P $+1 F '111' D. Q+ Find and Delete the second '111' record.  Check Cursor.
→55                 (Current Record)
D. Q+               Delete Current Record.  Check Cursor again.
→33                 (Current Record)
```

3.14.5.4 Record Group Output

Four commands will copy a group of Text Region records to the Editor's output files. Each command corresponds to a different file, but the parameter requirement is the same -- a Record Group specification, shown here as rgrp:

```
W rgrp      Write Primary Output to SO
O rgrp      Output Alternate to S1
L rgrp      List on LO
V rgrp      View on CO
```

The various ways of specifying a Record Group are described in section 3.14.5.1. For convenience, especially when using the V command, the Editor will accept any of these commands with no parameter at all. The result is the same as if a Period had been entered -- that is, the output will be the Current Record only.

```
W $#                Entire Text Region to SO.
O #-3               Last 4 records to S1.
L .+23              Current Record and 23 more to LO.
V                   Current Record only to CO.
P'33'VP'77'V        Move Cursor after next match on '33' and display record.
                    Move Cursor after next '77' and display record.
```

Record Group output commands have no effect upon the contents of the Text Region. Section 3.14.8 describes how the Editor's output files are opened and closed, and how dynamic buffer allocation interacts with Text Region and Save Region allocation.

If the Text Region is completely empty when one of these commands is entered, the command will fail, and the Editor will type out TEXT REGION EMPTY.

3.14.5.5 Formatting of List Output for Assembler Language

An L command -- List Record Group on LO -- ordinarily generates the same kind of records as any other command for output, except that the Operating System automatically inserts page breaks. If the text being listed is an Assembler Language source program, created free-form with the Text Editor, then an exact printed reproduction can be rather difficult to read. The Editor, like the Assembler itself, can be requested to spread the source statement fields into uniform columns.

Assembler Language formatting for LO is established when an L command is suffixed with a Plus Sign, before the Record Group parameter is entered. Formatting remains in effect until it is cancelled by an L command suffixed with a Minus Sign:

```
L+ $+2              Set LO for Assembler Language formatting.
                    List the first 3 records in the Text Region.
L #-2               List the last 3 records, still formatted.
L- $#               Restore unformatted output on LO.
                    List the entire Text Region.
```

3.14.5.6 Record Group Input

Two commands will read the Editor's input files, adding as many records as possible to the end of the Text Region. No parameters are involved:

```
R   Read Primary Input from SI.
A   Alternate Input from SA.
```

Either of these commands will automatically append enough new records to fill the available free space after the existing Text Region records. Reading will, of course, stop when an End-of-File is encountered; the Editor will type out END OF xx FILE.

After an R or A command, the Cursor will be under the first position of the first new record.

```
R V.+2 V#-2          Read more Primary Input.
                     View the first 3 new records.
                     View the last 3 new records.
D$# A V              Delete from the First Record thru the Last Record -- that is,
                       clear the Text Region.
                     Fill the empty Text Region from SA.
                     View the new First Record, which is also the new Current
                       Record.
```

Section 3.14.8 describes how the Editor's input files are opened and closed, and how dynamic buffer allocation interacts with Text Region and Save Region allocation.

3.14.5.7 Next Record Group

If the Primary Input on SI contains more records than the Text Region can hold, the usual technique is to read a piece of SI, edit the Text Region, write all the edited records to SO, clear the Text Region, and start again:

```
(editing commands)
(editing commands)
W $#                 Write the entire Text Region to SO.
D $#                 Clear the Text Region.
R                    Read enough of SI to refill the Text Region.
(editing commands)
(etc.)
```

This pattern is so common that a special Editor command -- N -- has been provided to "roll over" the Text Region. These two command lines are equivalent:

```
N
W$# D$# R
```

The N command can be used even when the Text Region is empty -- for example, just after the Editor has started execution. The effect will be the same as a simple R command.

After an N command, the Cursor will be under the first position of the First Record.

3.14.6 RECORD LOCATIONS

3.14.6.1 Record Location Parameters

Some commands accept a parameter which specifies a single record location:

I	Insert After Record
P	Position Cursor at Start of Record
U	Unsave After Record

For these commands, an acceptable parameter has one of these formats:

$	First Record
.	Current Record
#	Last Record
$+n	n records after First Record
.-n	n records before Current Record
.+n	n records after Current Record
#-n	n records before Last Record
$-1	Before First Record (Not acceptable for a P command)

Here are some examples of Record Locations:

I #	Insert after the Last Record.
U $-1	Copy Save Region to Text Region before First Record.
P $	Move Cursor to beginning of Text Region.
P $+3	Move Cursor to 3rd record after First Record.

3.14.6.2 Inserting Records

An Insert command -- I with a Record Location parameter -- copies new records from CI to any specified point in the Text Region. Adjustments are made to the identities of the First Record and the Last Record as needed.

After an Insert Records command, the Cursor is moved to the start of the final new record.

The Editor accepts the new records in any of the New Text formats described in section 3.14.4.8. The line-by-line format is used in these examples:

```
I. cr                 Insert after Current Record.
T? NEW RECORD cr      Editor asks for New Text.  One record entered.
T? ANOTHER cr         Editor asks for New Text.  Second new record entered.
T? 'cr                Record entry terminated.
V $+1                 View First Record and 1 more following.
A1A1A1                (Editor types out records)
B2B2B2
I $-1 cr              Insert before all existing records in Text Region.
T? ZYXW 987 cr        One record entered.
T? 'cr                Record entry terminated.
Q+                    Check Cursor.
←YXW 987              (Current Record)
V $+2                 View (new) First Record and 2 more following.
ZYXW 987              (Editor types out records)
A1A1A1
B2B2B2
```

3.14.6.3 Using the Save Region

Two commands are used to copy records back and forth from the Text Region to the Save Region. The Save command -- S -- needs a Record Group parameter for the contiguous records to be copied into the Save Region:

```
S.                  Save the Current Record only.
S $+1               Save the First Record and 1 more following.
S #-3               Save the last 4 records in the Text Region.
S $#                Copy the entire Text Region into the Save Region.
```

The S command does not affect the Cursor or the contents of the Text Region. Saved records are not cumulative -- each S command clears the Save Region before copying a new group from the Text Region.

If an S command has no parameter at all, the Save Region is cleared, but no new records are copied into it.

The Unsave command -- U -- copies all of the records in the Save Region back into the Text Region, just after any specified Record Location:

```
U .                 Unsave after the Current Record.
U $+4               Unsave after the 5th record in the Text Region.
U #                 Unsave after the Last Record.
U $                 Unsave after the First Record in the Text Region.
U $-1               Unsave before the First Record.
```

The U command does not affect the contents of the Save Region, which are still available for subsequent Unsaves.

After a U command, the Cursor will be under the first position of the last record just copied back to the Text Region.

Here is an example of how the Save Region can be used to move a group of records from one place to another in the Text Region:

```
V $#                Display the entire Text Region.
AAAAAAA             (Editor types out all the records)
BBBBBBB
CCCCCCC
DDDDDDD
EEEEEEE
P $+1 V             Move the Cursor to the 'B' record and check it.
BBBBBBB             (Current Record)
S.+1 D.+1 V$#       Save the Current Record and 1 more following.
                    Delete the Current Record and 1 more following.
                    Display the Text Region.
AAAAAAA             (Editor types out all the records)
DDDDDDD
EEEEEEE
U #-1               Unsave after Last -1 (the 'D' record).
Q+                  Check new Cursor position.
→CCCCCC             (Current Record)
V $#                Check final result.
AAAAAAA             (Editor types out all the records)
DDDDDDD
BBBBBBB
CCCCCCC
EEEEEEE
```

3.14.7 CHARACTER EDITING

3.14.7.1 Character Group Parameters

Three commands need a parameter which identifies a contiguous group of characters:

```
C   Change Characters
D   Delete Characters
I   Insert Characters
```

For these commands, an acceptable parameter has one of these formats:

```
+n          Current Character and n more following
-n          Current Character and n more preceding
null        Current Character only
'context'   First match against string, starting at Current Character
```

Null means that the command has no parameter at all. Here are some examples of Character Groups:

```
D +4                Delete the Current Character and 4 more following it.
C +3,'**'           Replace 4 characters with 2 characters.
I '.' , ' '         After the next occurrence of a Period, insert one Blank.
P.  D               Move the Cursor to the start of the Current Record.
                    Delete the (new) Current Character.
D '&&'              Delete the next occurrence of 2 consecutive Ampersands.
```

3.14.7.2 Character Modifications Controlled by Position

The current position of the Cursor can be used to control these commands:

```
C     , 'new text'   Change Current Character only
C +n  , 'new text'   Change Current Character and n more following
C -n  , 'new text'   Change Current Character and n more preceding

D                    Delete Current Character only
D +n                 Delete Current Character and n more following
D -n                 Delete Current Character and n more preceding

I     , 'new text'   Insert before Cursor Position
I +n  , 'new text'   Insert before Cursor Position + n positions
I -n  , 'new text'   Insert before Cursor Position - n positions
```

Valid formats for the New Text needed by the C and I commands are described in section 3.14.4.8. The C command will replace the character group with the New Text, which need not be the same length. The I command will insert the New Text at the location specified. A previous P command with no parameter can position the Cursor just after the last character in the Current Record, which allows insertion at the end of the record:

```
P$ P Q+              Position Cursor to start of First Record.
                     Position Cursor after end of Current Record.
                     Check Cursor.
ABCDE                (Current Record -- Cursor after end of record)
I ,'012'             Insert 3 characters just before Cursor Position.
V                    View Current Record.
ABCDE012             (Current Record)
P. C+4,'XY' V        Position Cursor to start of Current Record.
                     Change 5 characters into 2.
                     View Current Record.
XY012                Current Record.
P P-1 D V            Position Cursor after end of Current Record.
                     Move Cursor back 1 position, under last character.
                     Delete Current Character only.
                     View Current Record.
XY01                 (Current Record)

P. D+2 V             Position Cursor at start of Current Record.
                     Delete Current Character and 2 more following.
                     View Current Record.
1                    (Current Record)
```

After a C or I command, the Cursor will be under the first position to the right of the newly changed or inserted characters. If the last character in the record was affected, the Cursor will be after the end of the record, just as for a P command with no parameters.

```
D$#  I,'ABCDEF'      Clear the Text Region.  Insert 6 characters.
P. I,'XY' V Q+       Position Cursor at start of Current Record.
                     Insert 2 characters before Cursor.
                     View Current Record.
                     Check Cursor.
XYABCDEF             (Current Record)
XY←BCDEF             (Current Record, showing Cursor)
C+5,'0123'           Change Current Character and 5 more following into 4.
Q+                   Check Cursor.
XY0123               (Current Record -- Cursor after end of record)
```

A D command eliminates the specified number of characters, and closes up the rest of the record. The Cursor is moved just to the right of the affected area. As with a D command controlled by context, deleting the last character in a record will leave the Cursor under the rightmost remaining character, and eliminating the entire record will jump the Cursor backward to the start of the preceding record.

```
V #-1                View last 2 records in Text Region.
DEFGH                (Editor types out records)
IJKLMNO
P# P+1               Position Cursor to start of Last Record.
                     Move Cursor forward 1 position, under the J.
D+1 V Q+             Delete Current Character and 1 more following.
ILMNO
I←MNO
D-1 V Q+             Delete Current Character and 1 more preceding.
MNO
←NO
D+1 Q+               Delete the Current Character and 2 more following.
←EFGH                (Current Record)
```

3.14.7.3 Character Modifications Controlled by Context

The next occurrence within the Text Region of a specific character group can be used to control these commands:

```
C 'context' , 'new text'    Change
I 'context' , 'new text'    Insert After
D 'context'                 Delete
```

Valid formats for the New Text needed by the C and I commands are described in section 3.14.4.8. The C command will replace the first matching character group with the New Text, which need not be the same length as the Context String. The I command will insert the New Text immediately after the match against the Context String.

After a C or I command, the Cursor will be under the first position to the right of the newly changed or inserted characters. If the last character in the record was affected, the Cursor will be after the end of the record, just as for a P command with no parameters.

```
C 'LDA' , 'LDX'  Q+      Change the next occurrence of LDA into LDX.
                         Check Cursor.
PART3 LDX ADDR           (Current Record)
I 'ADDR' , ' COMMENT'    Insert After next match.
Q+                       Check Cursor.
PART3 LDX ADDR COMMENT   (Current Record -- Cursor after end of record)
```

A D command is equivalent to a C command with New Text of zero length. The remaining characters in the record are closed up over the deleted area. Again, the Cursor will be just to the right of the affected area. However, if the last character in the record is among those deleted, the Cursor will be under the new last character, not hanging after the end as with a C or I command.

```
P. D'RT3' Q+          Position Cursor to start of Current Record.
                      Delete first occurrence of RT3.
                      Check Cursor.
PA←LDX ADDR COMMENT   (Current Record)
D 'MENT' V Q+         Delete first occurrence of MENT.
                      View Current Record.  Check Cursor.
PA LDX ADDR COM       (Current Record)
PA LDX ADDR CO←       (Current Record, showing Cursor position)
```

A special situation arises when a deletion leaves no record at all. In this case, the Cursor is moved under the first character of the preceding record.

```
P $+4 V.              Make the 5th record in the Text Region the new Current Record,
                      and View it.
KLMN                  (Current Record)
V .-2                 View Current Record and 2 more preceding.
CDEF                  (Editor types out records)
GHIJ
KLMN
D 'KL' Q+             Delete KL and check Cursor.
←N                    (Current Record)
D 'MN' Q+             Delete MN and check Cursor.
←HIJ                  (Current Record)
```

3.14.8 SPECIAL FACILITIES

3.14.8.1 Command Loops

It is often useful to loop thru an Editor command, or a whole line of commands, making systematic access to the Text Region. An X Plus command -- the letter X followed by a Plus Sign, followed by a decimal number -- will repeat the single immediately following command until the count is exhausted.

```
X+20 C'DOG','CAT'    Execute the next command 20 times.
                     Change an occurrence of DOG into CAT.
X+4 I.,'* *'         Execute the next command 4 times.
                     Insert new record after the Current Record.
```

The command being repeated -- called the object of the X Plus -- is usually a Change, Insert, or Delete which automatically moves the Cursor for each repetition. The Editor rejects an attempt to use, as the object of an X Plus, either another X Plus command, or the X Minus command described next.

An X Minus command -- the letter X followed by a Minus Sign, followed by a decimal number -- will repeat the entire line currently in the Command Region. To enter a new line into the Command Region, an E command is used:

```
? E                  Enter line into Command Region.
E? P' JST ' L P.+1   Editor requests entry.  Command Region set...
                     Position Cursor after next JST op code.
                     List Current Record.
                     Position Cursor to start of next record.
? P$  X-100          Position Cursor to start of Text Region.
                     Execute Command Region 100 times.
```

An E command must be the last on its line. The entered line completely replaces the previous contents of the 72-character Command Region. Two commands are not acceptable within the Command Region -- an X Minus and an E. There is no restriction on the use of an X Plus within the Command Region.

Here is a more elaborate example of using the X and E commands. Suppose that we have an Assembler Language Source Program on punched cards or paper tape. The program was written in a hurry, and the statements lack any comments. We can take advantage of the fact that OS will transfer each card or paper tape record truncated after the rightmost non-blank character. That is, the position at which we want a Comments Field is after the end of each record, just where a P command with no parameter will put the Cursor.

For Source Programs maintained with the Text Editor, it is very convenient if each Comments Field starts with a Period, giving us a hook for a context scan. While we're appending comments to the records, we will make them look like this:

```
LABEL MNEMONIC OPERANDS .COMMENTS FIELD
```

We want a Command Region which will do all this:

1. Position the Cursor to the next record.
2. Position the Cursor after the end of the existing characters.
3. Type out the record, so we can decide what comments, if any, to append.
4. Automatically append one Blank and one Period, to start the Comments Field.
5. Accept New Text for the rest of the Comments Field.
6. Repeat the whole process.

```
? E cr                      Enter line into Command Region.
E? P.+1 P V I,' .cr         Editor requests entry.  Command Region set...
                            Position Cursor to start of next record.
                            Position Cursor after end of Current Record.
                            View Current Record.
                            Insert Before Cursor, 2 characters of New Text.
                            New Text still open.
```

Notice that the I command uses New Text in the open-ended mixed format. This forces us to put the I command at the end of the Command Region, and the P.+1 command at the beginning. The result is that the very first record in the Text Region will not participate in the command loop processing -- but it's probably a TITL statement anyway.

Now we read the Source Program, which has 117 records in it. The R command leaves the Cursor at the start of the program, and we loop thru the Command Region 116 times.

```
?R cr                       Read Primary Input from SI.
W38 END OF SI FILE          (Editor message)
? X-116 cr                  Execute Command Region 116 times.
GO STX OSRTN                (Second record in Text Region)
T?SOME COMMENTS'cr          New Text requested, entered, and terminated.
 SPACE 1                    (Command loop continues)
T?'cr                       No Comments Field supplied for this record.
ETC                         (Command loop continues)
```

At the end of the loop, we can check the second record:

```
? P $+1 V                   Position Cursor at start of Statement 2; view it.
GO STX OSRTN .SOME COMMENTS (Current Record)
```

3.14.8.2 File Handling -- Opening Blocked Files

The Text Editor accesses each of its 7 files thru standard IOCS techniques. Each file is automatically opened the first time it is needed, and not before. By deferring the allocation of blocking and deblocking buffers until they are actually required, a significant amount of extra space is usually available for the expansion of the Text Region and the Save Region.

However, the deferred opening of a blocked file does create one complication. If the Text Region and the Save Region are occupying all of the memory above the fixed part of the Editor, then no space is available for the dynamic allocation of new blocking or deblocking buffers, and no more blocked files can be opened. The Editor will issue the message INSUFFICIENT SPACE, and reject any command which would involve I/O for unopened blocked files.

To recover from this situation, some memory must be freed. If the Save Region is not empty, it must be cleared by an S command with no parameter. If it turns out that even more memory is needed, then some of the records in the Text Region must be deleted, perhaps after they are written out to an unblocked file, or to a blocked file which is already open.

This whole problem can be avoided very simply. If it's at all practical, tell the Editor explicitly to Open all blocked files before anything else is done. The command is a Colon -- to signify a special file-handling command -- followed by the letter O, followed by a Comma and a list of the Logical Unit Names to be forced open immediately:

```
:O,SI              Open the SI (Primary Input) file.
:O,SA,SO,S1        Open the files on SA, SO, and S1.
```

An Open command can be entered for an unblocked file as well, but the record area for such a file is allocated in the fixed part of the Editor, and has no connection with the free memory problem just described.

If the :O command is entered for a file which is currently open, an OS Close/Save request will be issued for the file, followed by an OS Open request.

3.14.8.3 File Handling -- Saving and Releasing Files

A normal termination of the Text Editor, thru the T command described in another section, will automatically issue a Close/Save for every input and output file. This means that each old input file, and each newly-created output file, is available for input to subsequent programs.

In special situations, it may be necessary to force a file closed without terminating the Editor. For example, an output file assigned to a paper tape punch could be given an intermediate EOF record; the next output command will automatically re-open the file. To explicitly close a file, and to allow its future use if it's on a magnetic device, a Close/Save command is used:

```
:S,SO              Close/Save the SO (Primary Output) file.
:S,SA,SO,S1        Close/Save the files on SA, SO, and S1.
:S,SO :O,SO        Close and re-Open SO.
```

- Another use for the :S command is shown in section 3.14.8.4, Suspending the Text Editor.

To explicitly close a file, and to delete a magnetic device file, a Close/Release command is used:

```
:R,SI              Close/Release the SI (Primary Input) file.
:R,SI,SA,S1        Close/Release the files on SI, SA, and S1.
```

A Close/Release of a magnetic device file has the same effect as a DE command for OS:CPY, and makes it unnecessary to execute OS:CPY to delete obsolete input or secondary files. For a file assigned to a non-magnetic device, either a :R or a :S command can be used; the effect is the same.

Closing a blocked file does not make its buffer space available for other uses. However, another Open of the same Logical Unit, with a block size no greater than before, will re-use the old buffer rather than force allocation of still another one.

3.14.8.4 Suspending the Editor

It is possible to escape temporarily from the Text Editor, and return control to the Operating System for re-assignments, time log-outs, and other special services. The Editor command is simply a Slash as the last entry on a line:

```
/                   Escape to OS.
W$# :S,SO /         Write entire Text Region to SO file.
                    Close/Save SO file.
                    Escape to OS.
```

The Editor issues a standard SPND: request. The next type-out on CO is from the Operating System, rather than from the Editor. After the necessary OS commands are entered, control is returned to the Editor with a /RESUME, and the Editor is ready for more processing:

```
?:R,SI :S,SO /          Close/Release SI.
                        Close/Save SO.
                        Escape to OS.
>/ASSIGN SI=SO          Use Primary Output just created as the new Primary Input.
>/ASSIGN SO=D0.SRCFL    Assign new Primary Output.
>/RESUME                Back to Text Editor.
?                       Editor ready for new command.
```

Observe that OS never allows a file to be open for creation, and open for reading, at the same time. The only way an SO or S1 file can ever be turned around for input to the Editor, or to any other program, is to be processed thru a Close/Save before it is used for an explicit Open (with a :O command) or an implicit Open (with an R, A, or N command).

3.14.8.5 Terminating the Editor

A normal Terminate command -- T -- simulates a loop of N commands until EOF is reached on SI, then simulates a Close/Save on all files. In detail, the sequence of events for a T command is:

W $#	1. Write the entire Text Region to SO.
D $#	2. Clear the Text Region.
R	3. Read enough of SI to refill the Text Region.
	4. Repeat from Step 1 until End-of-File on SI.
:S,SI,SO,SA,S1,LO	5. Close/Save all files.
	6. Issue standard termination request to Operating System.

If Step 1 conflicts with your requirements -- if, for example, you have already entered a W command, and don't want another copy of the Text Region written to SO, simply clear the Text Region before entering the T command:

```
D$# T          Clear Text Region and Terminate.
```

To suppress the whole SI-to-SO copy loop, and begin the termination procedures at Step 5, follow the T with a Plus Sign parameter:

```
T+             Close/Save all files, and exit to Operating System.
```

Here is an OS job which generates a formatted listing of an Assembler Language Source Program. It takes advantage of the fact that SO can be assigned to a line printer. The initial I command is needed to insert a dummy blank record in the Text Region. Otherwise, the L+ command would fail because of an empty Text Region, and the T command would not be processed.

```
/JOB

/ASSIGN LO=LP,SO=LP,SI=D0.SOURCE

/EXEC OS:EDT

 I.,' '   L+.   T

/NJOB
```

3.14.9 MESSAGES

3.14.9.1 Message Detail

Error Messages and Warning Messages from the Editor look like this;

```
E10     INVALID COMMAND
E13     INSUFFICIENT SPACE
W33     TEXT NOT LOCATED
W38     END OF SI FILE
```

When the message numbers become so familiar that the text is superfluous, a special command is available to drop the text from all Editor messages. The command is an M followed by a Plus Sign, and remains in effect until cancelled by an M command with no parameter:

```
J
E10 INVALID COMMAND
M+ J
E10
M J
E10 INVALID COMMAND
```

Another Editor command -- M followed by a Minus Sign -- will suppress Warning Messages entirely. This can be useful when the count parameter of an X Minus -- Execute Command Region -- is based on a rough estimate, and a number of superfluous messages might be generated once the loop has run past the end of the Text Region.

```
M- X-200 M          Suppress Warning Messages.
                    Execute Command Region 200 times.
                    Restore full message mode.
```

In M- mode, Error Messages are cut down to the message number alone, just as in M+ mode. Similarly, an M (or M+) command cancels the effect of an M- command.

3.14.9.2 Error Messages

E10 Invalid Command
A new command was expected, but the next non-blank was not a command.

E11 Invalid Argument
The parameter following a command was not in any recognizable format.

E12 Illegal Argument
The parameter was in a valid format, but cannot be used for this particular command.

E13 Insufficient Space
Not enough memory is available for the Text Region and the I/O buffers.

E14 Numeric Range Error
The value of n in a parameter exceeds the limit of 32767.

E15 Missing Closing Quote
Command line terminated in the middle of a Context String parameter.

E16 New Text Missing
Insert or Change command had a Comma not followed by a Single Quote or a Carriage Return.

E17 Illegal E Usage
E command in the Command Region, or E command was not last in a command line.

E18 Illegal X Usage
X- command in the Command Region, or just after an X+ command.

E19 No Entered Line
X- command, but Command Region is empty.

E20 Illegal Record Range
Attempted access to records beyond current limits of Text Region.

E21 Invalid Unit Name
Open or Close with parameter other than SI, SO, SA, S1, or LO.

3.14.9.3 Warning Messages

W30 Text Region Empty
Command referred to records which do not exist.

W31 Text Region Full
No room for more input to be appended.

W32 Save Region Empty
U command, but no records exist to unsave.

E33 Text Not Located
Context String parameter found no match. Cursor remains where it was before.

W34 Outside Text Region
Attempted access to records or characters beyond current limits of Text Region.

W35 Outside Current Record
Attempted character manipulation would have affected two records at once.

W36 Insufficient Space
Text Region or Save Region could not be expanded to the size needed.

W37 Record Too Large
Attempted output had too many bytes for device. Record was truncated and written.

W38 End of SI File
Primary Input has reached EOF.

W39 End of SA File
Alternate Input has reached EOF.

W40 String Too Large
New Text record exceeded limit of 80 bytes.

3.14.9.4 Information Messages

OS:EDT (nn)
Text Editor has started execution. Program version number is nn.

I/O CHECK
The Operating System has detected an error condition during I/O processing.
To continue execution of the Editor, enter /RESUME, else enter /CANCEL

OS:EDT END
Text Editor has ended execution after a T or T+ command.

3.14.10 TEXT EDITOR COMMAND SUMMARY

CURSOR

P	Position Cursor.
F	Find matching label.
Q	Display Current Record up to Cursor.
Q+	Display entire Current Record with Cursor.

EDITING

C	Change.
D	Delete.
I	Insert.

INPUT AND OUTPUT

R	Read Primary Input from SI.
A	Alternate Input from SA.
N	Next -- Text Region to SO, SI to Text Region.
W	Write Primary Output to SO.
O	Output Alternate to S1.
L	List on LO.
V	View on CO.

CONTROL

:O,	Open file.
:R,	Close/Release file.
:S,	Close/Save file.
E	Enter into Command Region.
X-n	Execute Command Region n times.
X+n	Execute next command n times.
M M+ M-	Messages full length. Messages by number only. Messages for Errors only.
S	Save records.
U	Unsave records.
/	Escape to Operating System.
T	Terminate, after copying Text Region and SI to SO.
T+	Terminate, without copying.

Section 4

PROGRAM/SYSTEM COMMUNICATION

4.1 INTRODUCTION

User programs communicate with the Operating System through a series of instruction sequences known as supervisor calls. These calls cause the system to perform Input/Output or other executive services for the program.

The general form of a supervisor call is:

```
LABEL     JST      External Name
          DATA     Address of Parameter List
           *       Busy or Error Return
           *       Normal Return
```

4.2 REQUESTS FOR INPUT/OUTPUT CONTROL SERVICES

All requests for I/O services are initiated by a call to an IOCS entry point. These entry points are declared external to the user program (using the EXTR or REF directives) and are resolved when the program is processed by OS:LDR or OS:LNK.

IOCS requires that each request be accompanied by the address of a parameter list, known as a control block, which describes the activity required. These control blocks (and any record, buffer or working areas required by them) are located in the user's program area.

All requests preserve the calling program's hardware state and registers, with the exception of the A register. IOCS returns, in the A register, the status and results of the program's request. The format and content of this register is identical to that of the "status" word of the file control block (FCB). The file control block is described in section 4.4.1.

In the discussion of I/O requests which follows it is generally assumed that the system will return control to the calling program after the requested operation has been completed. This is the normal method, and is known as the Auto/Wait mode of operation.

However, it is possible for the calling program to regain control after the operation has been initiated, perform other unrelated functions, and not wait for completion until such completion is actually required. Such a mode of operation is obtained by selecting the non-Auto/Wait attribute in the appropriate FCB and requires use of the WAIT: or TEST: requests.

4.2.1 The OPEN: Request

```
LABEL    JST     OPEN:
         DATA    FCB
           *     Busy/Error Return
           *     Normal Return
```

This request is required prior to the execution of any other operation on the file, and allows IOCS to perform initialization and linkage generation between the system and the user's file control block (FCB).

When IOCS receives the OPEN: request, it verifies (1) that the required assignment of the physical device has been made, (2) that any file names required are present, and (3) that the device is ready for data transmission. Any error causes the system to reassign itself to the operator console, issue an appropriate message, and then suspend itself to allow remedial action. After the operator has corrected the problem, he may enter a /RESUME command. IOCS will automatically re-issue the OPEN: request. A successful Auto/Wait OPEN: will always take the "normal" return; an "error" return probably indicates a wrongly coded FCB.

If an OPEN: request is issued for a non-Auto/Wait file, and a previous OPEN: is still pending for the same physical device (for example, for another file on the same disk), the "busy/error" return will be taken. The user should go thru a WAIT: before re-trying the OPEN: of the file.

If the /ASSIGN command for a file-oriented device included a specific file name (for example, /ASSIGN S1=D1.XYZ), then this name will be inserted into the user's FCB (words 4 thru 6) by the OPEN: process, destroying any name previously stored there. The user must not alter the file name in an open FCB.

When an existing file is opened, IOCS will verify that the FCB does not conflict with the attributes of the file. If the FCB has allowed for a larger block size or record size than an input file really needs, the more accurate smaller values will be inserted into the FCB.

4.2.2 The CLOSE: Request

```
LABEL    JST     CLOSE:
         DATA    FCB
           *     Busy/Error Return
           *     Normal Return
```

This request is required after all I/O operations have been completed, so IOCS may terminate processing the file. Failure to issue a CLOSE: request can cause user data to be lost.

When IOCS receives the CLOSE: request, it performs any I/O operations still required (last block output, etc.) and severs the system-to-FCB linkage.

IOCS will return to the user through the "normal" exit if the request was successful. If it was not possible to CLOSE: the device, the "busy/error" return will be taken. If this condition goes uncorrected, data may be lost and the system may be unable to use the file at a future time.

4.2.3 The IO: Request

```
LABEL       JST         IO:
            DATA        IOB
             *          Busy/Error Return
             *          Normal Return
```

The IO: request allows the program to perform data transmission (READ and WRITE) and special processing such as REWIND, BACKSPACE, and EOF. The user indicates to IOCS the operation requested, and any record addresses or character counts required, in the Input/Output Control Block (IOB). (See section 4.4.2 for the format of an IOB.) IOCS returns the result of that request as information in the related File Control Block (FCB).

When IOCS receives an IO: request, it verifies that the file has been successfully opened, that the device is physically available, and that the request is valid for that device. Any error will cause an operator message and an error return.

The request is then passed on to the appropriate driver, which performs the actual operation and returns to IOCS any errors encountered. Finally, IOCS prints any error messages required and returns to the calling program through the applicable return. In all cases, the current status will be returned in the A register and the FCB.

4.2.4 The WAIT: Request

```
LABEL       JST         WAIT:
            DATA        FCB
             *          Busy/Error Return
             *          Normal Return
```

The WAIT: request is used when the non-Auto/Wait I/O mode of operation is selected. It allows the calling program to issue an IO: request, regain control, and test for I/O completion at a later time. A WAIT: request suspends execution of the calling program until the I/O operation has completed, and returns the status of that operation in the A register and the FCB.

When IOCS receives a WAIT: request, it retains control until the device indicates that it has completed its operation or an error has occurred. An error will cause an operator message, and the calling program's error return will be taken. Successful completion will result in a normal return to the calling program.

4.2.5 The TEST: Request

```
LABEL       JST         TEST:
            DATA        FCB
             *          Busy/Error Return
             *          Normal Return
```

The TEST: request allows the program to interrogate the status of a file or device at any time and continue processing accordingly. The user may thus query the system for I/O completion, as with the WAIT: command, but receive control back regardless of status.

When IOCS receives a TEST: request, the current status of the unit is immediately returned to the calling program in the A register and the FCB. The unit's status will determine the return taken and error messages will be output to the operator as required.

4.3 REQUESTS FOR EXECUTIVE SERVICES

All requests for executive services are initiated by a call to a system entry point. These entry points are declared external to the user program (using EXTR or REF directives) and are resolved when the program is processed by OS:LDR or OS:LNK. Executive requests require a parameter list or message located in the user's program area. All executive requests restore the calling program's previous hardware state and registers on return, unless otherwise specified in a request description.

4.3.1 The SUPV: Request

```
LABEL       JST         SUPV:
            DATA        SRB
             *          Return
             *
SRB         DATA        Request Code
            DATA        As Required
```

The SUPV: request provides a variety of executive services. A System Request Block (SRB) of four or more words in the calling program is used as an interface. The low-order byte of word 0 of the SRB -- the Operation Code (OPR) -- determines the specific request, and is never altered by the system. The high-order byte of word 0 -- the Event Control Byte (ECB) -- is set to zero each time the system receives a SUPV: request, and is altered only to indicate an unsuccessful request. For example, an invalid value for OPR will return with OPR unchanged, and the ECB set to :80. The content of the remainder of the SRB before and after the request is determined by the value of OPR, as described in the following sections.

1 -- Request Free Memory Limits

The system returns, in the SRB, the boundaries of currently unused scratchpad and main memory. All locations between, and including, these boundaries are not occupied by OS nor by the user program itself, and may be used for building tables or code as needed. The SRB will contain:

Word 1	Low boundary of free main memory
Word 2	High boundary of free main memory
Word 3, Byte 0	Low limit of free scratchpad
Word 3, Byte 1	High boundary of free scratchpad

2 -- Request Current Date

The system returns, in the SRB, the exact six ASCII characters which were entered between slashes in the most recent /DATE command. The user is reminded that these are not necessarily numeric characters; nor is it true that the first two characters (for example) do, or do not, represent the current month. Each installation sets its own standards for /DATE commands. The six characters are returned contiguously in SRB words 1, 2, and 3.

3 -- Request Current Time in ASCII

The system returns, in the SRB, the current real time, converted to ASCII numerals. Words 1, 2, and 3 contain the hours, minutes, and seconds, respectively. The accuracy of this information is wholly dependent upon the accuracy of the most recent /TIME command.

4 -- Request Current Time in Binary

The system returns, in the SRB, the current real time, converted to three separate 16-bit binary values. Words 1, 2, and 3 contain the hours, minutes, and seconds, respectively. The accuracy of this information is wholly dependent upon the accuracy of the most recent /TIME command.

5 -- Request the Next Parameter

The system returns, in the SRB, the byte address and the length of the next available program parameter, exactly as entered on an /EXEC, /BEGIN, or /RESUME command. Each individual parameter is defined as all the characters bounded by, but not including, a following comma or line terminator. Each of these examples has three parameters:

```
/EXEC      PROGX,MM=5000,NL,TERM
/BEGIN     ,MM=5000,NL,TERM
/RESUME    MM=5000,NL,TERM
```

The parameters in each example have lengths of 7, 2, and 4, respectively. Each successive request for the next parameter would return the byte address and length of just one of these.

Word 1 of the SRB will contain a byte address, and word 2 a byte count. When a request is made for the next parameter, but none exists, the system will return a byte count of zero. If the program still is not satisfied, and makes another request, the system will request more parameters from the command input (CI) unit. If CI is assigned to the operator console, "?" will be displayed on the command output (CO) unit to indicate that a response is required. The program may continue to request parameters from the system, and CI may be used to supply them, as often as required.

6 -- Request Operator Communication

The system requests a response from the operator, optionally preceded by a message from the program. The message, if one is supplied, is displayed on the Command Output (CO) unit. If the CI unit is a console, the system will always display "?" on the CO unit, to indicate that a response is required. The operator responds on the Command Input (CI) unit.

The program must supply, in word 3 of the SRB, either the word address of the message desired, or a value of zero, indicating that no message is involved. The message must be in the format described in the section on the MSG: Request. Words 1 and 2 of the SRB will contain the byte address and byte count of the response.

7 -- Request Physical Device Information

The system returns, in the SRB, certain information about the physical device currently assigned to a given logical unit. This information is used by programs which determine I/O techniques at execution time, based upon the Unit Control Block (UCB) for each physical device.

The program must insert into word 1 of the SRB a pair of ASCII characters representing a valid logical unit name -- for example, LO or S2. Upon return to the program, a request for a non-existent or unassigned logical unit will be indicated by setting word 0 of the SRB to a value of :8007. For a valid request, word 0 still will be :0007. Word 3 of the SRB will contain the maximum bytes for a binary record on the physical device. The low-order byte of word 2 will contain the maximum characters for an ASCII record. If the device is appropriate for page control, the high-order byte of word 2 will contain the maximum lines per page; otherwise, zero.

8 -- Request Loading of a Program Overlay

The system loads into memory a program designated by name. The program must be suitable for /LOAD command processing, and require the resolution of no external references at all.

This request needs a five-word SRB, rather than the usual four words. The user must set word 0 of the SRB to a value of :0008, and words 2, 3, and 4 to the ASCII name of the program (with trailing spaces as needed). The exact same name must appear in the directory of the device currently assigned as the System File (SF) unit.

If the program is relocatable, word 1 of the SRB must contain the relocation bias. If it is absolute, word 1 is ignored. The requested load must fall completely within the User Area of memory, and must not overlay the request coding sequence itself, not its related SRB.

Control is never passed by the system to the newly loaded program, but remains in the original calling program. On return, both the A register and word 0 of the SRB indicate the success of the load request. This indication is a value of :nn08, in which "nn" is the new value of the ECB (and the high-order byte of the A register) as follows:

:00	Successful load
:81	I/O error on SF
:82	Load error, including violation of memory boundaries
:83	Illegal type code

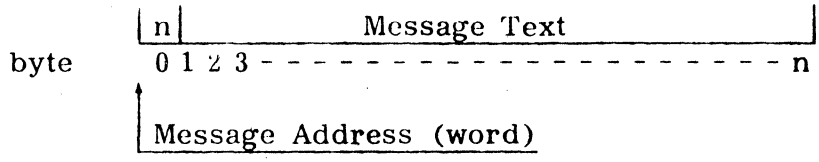
4.3.2 The MSG: Request

```
LABEL    JST     MSG:
         DATA    Message Address
           *     Return
```

The MSG request permits the calling program to print a message on the command output device (CO) and continue normal processing. No operator response is expected or allowed.

The message to be printed resides in the calling program, and its word address is passed in the calling sequence. The message can be up to 254 characters long; all valid ASCII characters are allowed. The first character position (byte 0) must contain the message length, in bytes. Any required line terminator, such as carriage return/line feed, will be added by the system and should not be included.

Message Format:



4.3.3 The SPND: Request

```
LABEL    JST     SPND:
         DATA    Message Address
           *     Return
```

This request allows the calling program to print a message for the operator, and then suspend itself for operator action.

The SPND: request causes the system to reassign the command input (CI) and command output (CO) devices to the operator console, after which the message is printed on CO. The message passed must be in the format described for the MSG: request.

The operator may return control to the program with the /RESUME command. This will cause restoration of the original CI and CO assignments and all program status.

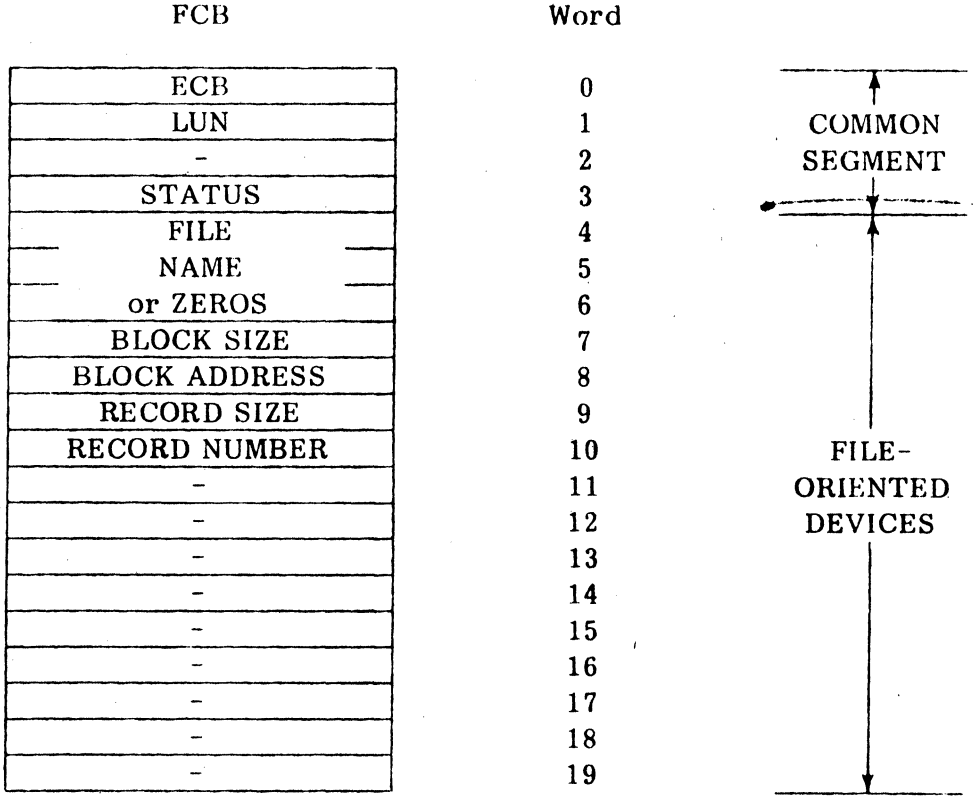
4.3.4 The TERM: Request

```
LABEL    JST     TERM:
```

The TERM: request is the last instruction executed in the user's program. It returns control to the system, allowing program termination in an orderly manner. The programme may use either a JST or a JMP instruction for this request.

4.4 IOCS CONTROL BLOCKS

4.4.1 The File Control Block (FCB)



The file control block (FCB) is a 4 or 20 word list supplied by the user program. An FCB is required for every file which the program references, and describes to the system the attributes of that file. The common segment (words 0-3) is required for all files; the extended segment (words 4-19) is required for those on file-oriented devices (magnetic tape, disc, cassette). It is recommended that the extended form generally be used, to facilitate future file-oriented device operation without program modification.

The user is required to initialize or examine the entries of the FCB labelled in the illustration. The unlabeled entries are required by the system. They should be initialized to binary zeros and left undisturbed.

4.4.1.1 ECB - Event Control Block

The ECB describes the attributes of the file, the type of processing required, and its final disposition. This entry must be initialized by the user and should not be altered after the file is opened. Each bit off (0) is the default condition.

Bit 15 0 = Auto/Wait. IOCS returns control after an IO: request has been completed.

1 = Non-Auto/Wait. IOCS returns immediately after an IO: request has begun. Used with WAIT: or TEST: request.

Bit 14 0 = Sequential IO processing. Standard on all devices.

1 = Random IO processing. Available on disc only.

Bit 13 0 = Physical record I/O. Causes a record to be read/written each I/O request. Standard on all devices.

1 = Blocked record I/O. Causes the physical records to be blocked (combined into larger groups) for each data transfer to allow greater device utilization. Available on all file-oriented devices.

Bit 12 0 = Keep the file after CLOSE.

1 = Delete (do not keep) the file after CLOSE.

Bit 11 0 = Must be off if either bits 10 or 9 are on.

1 = File OPEN for READ. If a file-oriented device, the file must currently exist. WRITE is not allowed.

Bit 10 0 = Must be off if either bits 11 or 9 are on.

1 = File OPEN for WRITE (creation). If a file-type device, the file must not currently exist. READ after WRITE is allowed.

Bit 9 0 = Must be off if either bits 11 or 10 are on.

1 = File OPEN for UPDATE (modification) by physical record number. Applies to RANDOM files and is available on disc only.

Bit 8 0 = Automatic page control. See Note below.

1 = Automatic page control suppressed. User is responsible for maintaining line counts and page separations.

Bit 7 0 = Automatic top-of-form when a list file is open or closed. See Note below.

1 = No automatic top-of-form when a list file is open or closed.

Bits 6--0 Current line number for a list file. Should be initialized to zero. See Note below.

NOTE: List Files

List files -- that is, files assigned to a teletype or line printer -- require special handling by IOCS because the user generally wants output divided into pages. The number of lines on a page is determined by word 5 of the Unit Control Block for each physical device, which ordinarily contains a value of 66, but may be altered during system generation.

If bit 8 is off, IOCS will update the line number in bits 6 through 0 of the ECB each time a line is output to a list file. A line number within 12 lines of the bottom of the page will force a top-of-form, and the line number will be reset to one.

If bit 8 is on, suppressing automatic page control the line number will always be 1.

Top-of-form is also forced once when a list file is opened, and once when it is closed, unless the user sets on bit 7 of the ECB. Normal programming practice for a list file is thus to zero bits 8 thru 0 of the ECB before the file is opened, and allow IOCS to handle all details of page control.

4.4.1.2 LUN - Logical Unit Name

The LUN contains the logical unit name of this file: two ASCII characters, taken from the available list as described in appendix D. Use of the /ASSIGN command will relate this name to the physical device required.

4.4.1.3 STATUS Word

This word contains the current unit and device status of the file referenced. It is updated after every call to request I/O and related services. This word must be zeroed prior to a file OPEN: request and must not be altered thereafter by the user.

Each 'on' bit of the status word indicates either general file information (the file is OPEN, etc.) or error information (data error, etc.). All error information is reset (bit set 'off') by the system at every call; the user need not clear these bits.

Bit	Meaning
15	Error occurred; specified in bits 14 thru 4.
14	Missing or invalid file name in FCB.
13	Unused bit position.
12	Device is not ready, or a file-oriented device is unlabeled.
11	Device is busy.

Bit	Meaning
10	Device error has occured.
9	Data error has occurred; data transferred has doubtful validity.
8	Illegal operation requested.
7	Duplicate file name found on OPEN.
6	Multiple concurrent WRITE not supported on this device.
5	File not found on OPEN.
4	Write-protect violation occurred.
3	End of medium was found.
2	End of file was found.
1	Successful OPEN occurred.
0	File-oriented device.

4.4.1.4 File Name

When operating with file-oriented devices (magnetic tape, cassette, disc), a file name must be made available to the system at OPEN time. This name can be placed in the FCB, or entered as an optional argument on the /ASSIGN command. The name consists of one to six alphanumeric characters and colons, the first of which must be alphabetic, with trailing spaces as required.

If a name is not placed in the FCB, the bytes must contain binary zeros. If a name exists in the FCB and one is entered on the /ASSIGN command, the ASSIGN name replaces the original name when the FCB is opened. If a name is not entered in either way, the system will request one.

4.4.1.5 Block Size

If blocking is requested (see ECB), this word must contain the total block length in bytes, computed as the product of the record size and the number of records per block. If unblocked, the word should contain binary zeros.

4.4.1.6 Block Address

If blocking is requested (see ECB), this word must contain the starting word address of a block buffer located in the user's area. The buffer need not be initialized. This word should be cleared to binary zeroes if unused.

4.4.1.7 Record Size

This word is required for all file-oriented devices and must contain the logical record length in bytes. This is the "defined" record length only; the actual count required is placed in the IOB at I/O request time and may be less. It may not be greater.

4.4.1.8 Record Number

If random access processing is requested, this word contains the logical record number of the record being processed. It is the responsibility of the calling program to maintain this word during I/O operations. The system will not update it in any way.

4.4.2 The Input/Output Control Block (IOB)

IOB	Word
OPR	0
FCB ADDRESS	1
RECORD LENGTH (bytes)	2
RECORD ADDRESS	3
TRANSFER COUNT (bytes)	4

The Input/Output control block (IOB) is a 5-word argument list supplied by the user's program. An IOB is required each time an IO: request is made. Normally a single IOB per file is sufficient.

The user must initialize words 0 thru 3 prior to an IO: request. The system will return the actual transfer count in word 4, but will not modify any of the other words.

4.4.2.1 OPR - Operation Code

The OPR field describes to the system the type of operation required. The following codes are valid:

1 Read symbolic (ASCII)

2 Read binary

3 Write symbolic (ASCII)

4 Write binary

5 Rewind the file

6 Backspace one record

7 Verify after write (file-oriented devices only)

8 Write EOF (See section 4.5.1)

4.4.2.2 FCB Address

This word contains the address of the FCB associated with this file.

4.4.2.3 Record Length

This word contains the length of the logical record in bytes. During a WRITE operation, it determines the length of the record to be output. During a READ operation, it determines the maximum number of bytes to be input. READing a record of greater length will cause all surplus bytes to be discarded.

4.4.2.4 Record Address

This word contains the starting word address of the record buffer located in the user's area, into which the record will be read, or from which it will be written.

4.4.2.5 Transfer Count

After completion of an I/O operation, this word will contain the byte count of the number of characters actually transferred. This number may not correspond to the Record Length contents, but can never be greater. If the transfer count is less than the requested amount, no fill characters are appended by the system.

4.5 DEVICE DEPENDENT CONSIDERATIONS

Although the Operating System supports all devices in a "device independent" manner, the user should be cognizant of certain hardware differences (such as various end-of-file indicators), as described in the remainder of this section.

4.5.1 End-of-File (EOF) Indicators

The system recognizes or generates various EOF indicators dependent upon the devices involved:

1. Teletype and HSPT devices: A separate record containing :FF0000 (rubout-null-null) in frames 1, 2, and 3. To generate EOF through the teletype keyboard, depress the rubout key once, then the SHIFT-CTRL-P keys twice. The sequence slash-asterisk (/*) is not recognized as EOF.

2. Card Reader: A card containing the two characters /* in column 1 and 2.

3. Magnetic Tape: A special record recognizable only to the hardware interface.

4. Cassette: A two-byte record containing :0000. This is a software end-of-file, and user programs should not generate such a record.

5. Disk: An internal software end-of-file recognizable only to OS.

4.5.2 Checksums

The system checks the accuracy of all binary paper tape records by the use of a checksum, as described in the Software Manual. Checksum calculations are not performed on symbolic paper tape records, nor on any other system devices.

4.5.3 Carriage Control of Printed Output

When writing to a "print" type device, such as the Teletype or line printer, the user controls page formatting through the use of control characters. A carriage control character is placed in the first byte position of the record to be printed and is included in the count of characters in that record. The control character is required by the driver and is not printed.

The valid control characters are described below; any other character will be treated as a "blank" and cause single spacing.

blank	Blank (or space) character. Single space before printing.
+	Plus character. No line spacing. Allows overprinting of the previous line.
1	Numeric One character. Advance to top-of-form before printing the line.
0	Numeric Zero character. Double space before printing.

NOTE

For the Teletype the top-of-form option in the Root may be used to cause 3 Linefeeds to be output in place of a top-of-form.

4.5.4 Recording Medium Preparation

Cassette cartridges and disk packs must be initialized before use under OS.

All cassette tapes used by OS must be formatted by the Cassette Address Formatter (Program 96066). After the cassette has been formatted, the write-enable tab for the address track (track A) must be removed. OS will not perform I/O on a cassette which still has its address track enabled for writing.

All disk packs used by OS must be formatted by one of the following programs:

Moving Head Disk Formatter Program (ID 96080)
Moving Head Disk Diagnostic Program (ID 96075)

4.6 PROGRAMMING EXAMPLE: IOCS AND EXECUTIVE REQUESTS

The assembly listing reproduced on the following pages will demonstrate a typical approach to coding IOCS and Executive requests. The program is a simple "80/80 list," with input on SI and output on LO.

Certain points about the program coding will be discussed here, not because the techniques are sophisticated, but because they are so typical of almost any program intended to run under OS.

All of the named entries within OS to which control will be transferred are declared as external references before the executable code. The object code for each JST shows that these references are unresolved at assembly time; the program will have to be processed thru OS:LNK or OS:LDR to fill in the correct indirect links thru scratch-pad. Individual REF directives for each external name could have been used as an alternative technique.

Each OPEN: and IO: request is followed by a provision for an error condition. The IO: request for input also needs a test for End of File. Each CLOSE: request has a NOP for an error condition during closing, because the nature of the files being processed does not require anything more elaborate.

Each IOB and FCB is established with a single DATA statement. The Record Length field of the output IOB is not fixed at assembly time; instead, the program tries to minimize peripheral transfer time by writing only the number of bytes actually read in.

The input and output buffers occupy the same area of memory, except that the output buffer must allow for a carriage control character, plus one more space to fill out the first word.

The data referenced in the MSG: request begins at ERRMSG, but is coded in two statements. If the message were altered during program execution, the TEXT statement would probably be given a separate label.

```
PAGE  0001  MM/DD/YY  01:30:05  IOCS AND EXEC REQUEST DEMONSTRATION
MACRO2 (A1) SI= OSUSER  BO=

0002                                EXTR   OPEN:,CLOSE:,IO:,TERM:,MSG:

0004        0000          START   EQU    $
0005  0000  F900  0000              JST    OPEN:
0006  0001  001E                    DATA   RDFCB      OPEN SI
0007  0002  F217  001A              JMP    ERROR
0008  0003  F900  0000              JST    OPEN:      OPEN LO
0009  0004  0027                    DATA   WTFCB
0010  0005  F214  001A              JMP    ERROR

0012  0006  F900  0000  READ        JST    IO:        READ 1 RECORD
0013  0007  0022                    DATA   RDIOB
0014  0008  2091  001A              JAM    ERROR
0015  0009  8213  001D              AND    EOFMSK     ISOLATE EOF BIT
0016  000A  3108  0013              JAN    END        EOF?
0017  000B  B21A  0026              LDA    RDIOB+4    TRANSFER BYTE COUNT
0018  000C  0150                    IAR               ADD 1 FOR CARRIAGE CONTROL
0019  000D  0150                    IAR               ADD 1 FOR LEADING BLANK
0020  000E  9A1E  002D              STA    WTIOB+2    SET OUTPUT RECORD SIZE

0022  000F  F900  0000  WRITE       JST    IO:        WRITE 1 RECORD
0023  0010  002B                    DATA   WTIOB
0024  0011  F208  001A              JMP    ERROR
0025  0012  F60C  0006              JMP    READ       END MAIN LOOP

0027        0013          END     EQU    $          EOF PATH
0028  0013  F900  0000              JST    CLOSE:     CLOSE SI
0029  0014  001E                    DATA   RDFCB
0030  0015  0000                    NOP               ERROR IN CLOSE
0031  0016  F900  0000              JST    CLOSE:     CLOSE LO
0032  0017  0027                    DATA   WTFCB
0033  0018  0000                    NOP               ERROR IN CLOSE
0034  0019  F100  0000              JMP    TERM:      REQUEST TERMINATION

0036        001A          ERROR   EQU    $
0037  001A  F900  0000              JST    MSG:       ISSUE MESSAGE ON CO
0038  001B  0059                    DATA   ERRMSG
0039  001C  F609  0013              JMP    END
```

Programming Example: IOCS and Executive Requests
Part 1 of 3

```
PAGE  0002  MM/DD/YY  01:30:05  IOCS AND EXEC REQUEST DEMONSTRATION
MACRO2 (A1) SI= OSUSFR  BO=

0041  001D 0004        EOFMSK DATA   :0004     EOF BIT IN FCB

0043  001E 0800        RDFCB  DATA   :0800,'SI',0,0
      001F D3C9
      0020 0000
      0021 0000
0044  0022 0001        RDIOB  DATA   1,RDFCB,80,RDBUFF,0
      0023 001E
      0024 0050
      0025 0031
      0026 0000

0046  0027 0400        WTFCB  DATA   :0400,'LO',0,0
      0028 CCCF
      0029 0000
      002A 0000
0047  002B 0003        WTIOB  DATA   3,WTFCB,0,WTBUFF,0
      002C 0027
      002D 0000
      002E 0030
      002F 0000

0049  0030 A0A0        WTBUFF DATA   :A0A0     CONTROL PLUS 1 BLANK
0050  0031 0000        RDBUFF RES    40,0

0052       0008              LIST   :08       SHORT TEXT

0054  0059 19A0        ERRMSG DATA   ERREND-$*2-1%8+' '
0055  005A C5D2              TEXT   'ERROR -- RUN TERMINATED '
0056       0066        ERREND EQU    $

0058       0000              END    START

0000  ERRORS
0000  WARNING
```

Programming Example: IOCS and Executive Requests
Part 2 of 3

```
PAGE   1     MM/DD/YY  01:45:34  OS:LNK (80) MEMORY MAP

PROGRAM
  MSG:   049B      TERM:  0684      OPEN:  0A05      IO:    0B14
  CLOSE: 0B41

MEMORY USAGE
  SCRATCH-PAD LITERAL     00F7-00FB
  MAIN MEMORY PROGRAM     1074-1DD9
  EXEC ADDRESS            1074

PROCESSED LSI 2 OBJECT

NO ERRORS
```

Programming Example: IOCS and Executive Requests
Part 3 of 3

Section 5

FILE MANAGEMENT SERVICES

5.1 INTRODUCTION

The File Management module of the Operating System provides directory maintenance facilities for the file-oriented devices (magnetic tape, disk, cassette). The user is able to access program and data files by name, without regard to the physical characteristics of the device.

All requests to file-oriented devices are made through IOCS, in a manner identical to other devices. It is, however, necessary that the user supply some additional information in the File Control Block (FCB) prior to file OPENing.

5.2 FILE ORGANIZATION

Every file-oriented device under OS contains a directory which describes, by name, all data and program files resident upon it. This directory is created by the OS:LBL utility program and maintained as required by the File Management routines.

The directory is contained in the first few records of the physical volume (reel of magnetic tape, disk pack, cassette cartridge) on the device. The exception to this is the system-residence unit, on which the directory follows immediately after the system file.

The directory starts with an entry describing the volume, called the Volume Table of Contents (VTOC). This entry contains pertinent information for OS, as well as user information such as creation date and name. The Volume Table of contents contains enough room for 320 disk file entries, 64 floppy file entries, or 160 mag tape files.

The remainder of the directory is segmented into File Description entries, one for each file on that volume. An entry contains the file name, creation date and time, and system information such as record and block size, and total file length.

The VTOC, the File Description entries, and the contents of any file can be displayed with the OS:VEW utility.

5.3 FILE ACCESS METHODS

File Management provides both sequential and random access capabilities to the user program. The sequential access method is the standard mode of data transfer; random access must be explicitly requested through the File Control Block (FCB) at OPEN time.

5.3.1 Sequential Access

Sequential file processing is available to the user on all file-oriented devices. Sequential files are accessed by logical record and automatic blocking/deblocking of records is available.

Sequential files are uniquely ordered: given record n, the next READ request will always return record n+1. It is possible to access previous records with the BACKSPACE operation, and to return to the first record of the file with the REWIND operation.

A READ or WRITE request automatically advances the file to the next logical record. Thus, to access record n-1 after READ or WRITE record n, the user must issue 2 BACKSPACE operations.

File Management provides automatic blocking and deblocking of logical records under sequential access I/O. All I/O requests access a single logical record. Its relative position in the physical records contained within the file is controlled by the File Management routines.

The user provides a record buffer and a block buffer in his program area, and the size and address of each in the appropriate FCB and IOB. The block buffer should be a multiple of the record size. A buffer whose length is not a multiple will waste the surplus area.

5.3.2 Random Access

Random access file processing is available only for disk devices. Random files are accessed by physical record; automatic blocking/deblocking is not provided.

Each record of a random file occupies exactly one sector of the disk. The first record within a specific file is Relative Record 1, the next is Relative Record 2, and so on. The user must set the Record Number field of the FCB to indicate which Relative Record is to be accessed on the next IO: request. Except during file creation, the FCB must always indicate OPEN for UPDATE.

The number of data bytes contained in each record is fixed at two bytes (or one word) less than the physical capacity of a sector. For a WRITE, the user sets the byte count in the Record Length field of the IOB. When the same record is accessed with a READ, the Transfer Count field of the IOB will contain the same byte count.

A random file must be created in a special way, so it can be defined to have a certain maximum Relative Record Number. The FCB must be set to indicate OPEN for WRITE, as opposed to OPEN for UPDATE. The proposed maximum Relative Record Number must be placed in Word 9 of the FCB. This word ordinarily contains the Record Size of a sequential access file. Word 10 of the FCB, Record Number, is not used during creation of the file.

The program attempts to OPEN the FCB. If enough space is available on the disk for the requested number of sectors, a normal return is taken, and the program issues a CLOSE. Subsequent use of the file will require an FCB set for UPDATE, as previously described. If enough space is not available, the End of Medium flag will be set in the Status Word of the FCB (Word 3, Bit 3), and an error return will be taken. The program must then decide whether to abort processing or make a smaller request for disk space.

Section 6

SYSTEM GENERATION

6.1 INTRODUCTION

The Operating System is delivered as a configured system on paper tape, including a "Root" module which may be altered by the user to meet changing requirements. The system supports a wide range of hardware options, and will meet the software environment needs of most ALPHA-LSI users.

6.2 HARDWARE CONFIGURATIONS

6.2.1 Minimum Hardware Requirements

An ALPHA-LSI computer with 16K of memory and the following processor-mounted options:

Teletype interface for ASR-33
Power Fail/Restart
Real Time Clock

One ASR-33 Teletype, or its equivalent

6.2.2 Additional Hardware Supported

High Speed Paper Tape Readers
High Speed Paper Tape Punches
Card Readers
Line Printers
Disks
Cassettes
Magnetic Tapes

An Appendix to this publication lists the makes and models of peripheral devices currently supported under OS.

6.3 DELIVERED SOFTWARE

The Operating System is delivered as a collection of paper tape modules, as shown here.

ID Number	Description
96530-41	DOS Complete System (absolute binary)
96530-21	OS Root (source)
96530-31	OS Root (object)
9653X-3X	OS Nucleus (library object)

	System Utility Programs		Copy to SF	Link to SF
96540-30	Label Files	OS:LBL	X	
96541-30	Copy Files	OS:CPY	X	
96542-30	Loader	OS:LDR	X	
96544-30	Dump	OS:DMP	X	
96545-30	View Files	OS:VEW	X	
96546-30	Debug	OS:DBG	X	
96547-30	Source File Editor	OS:SFE	X	
96548-30	Concordance	OS:CNC	X	
96549-30	Link Editor	OS:LNK	X	
96550-30	Text Editor	OS:EDT	X	
96551-30	Independent Loader	OS:ILD	X	
96554-30	Paper Tape Header	OS:HDR	X	
	Assemblers			
96543-30	No Macro Facility	OS:ASM		X
96552-30	With Macro Facility	MACRO2		X
96553-30	LSI-3 Programming	MACRO3		X

The following documentation is also included:

96530-00	OS User's Reference Manual
96530-10	DOS Root assembly listing
96530-51	DOS Root load map

6.4 SYSTEM GENERATION PROCEDURES

6.4.1 Configuration of the Operating System

The contents of the OS Root Program (ID 96530) determines the nature of a specific system. The inclusion of the required device drivers, interrupt locations, logical unit default assignments, and other system parameters, are all provided for by assembling OS Root.

The standard OS Root for LSI-2/10 (DOS10 and DOSIOE) provides for the following peripherals and default assignments:

Device	Physical Unit	Default Logical Unit
Teletype Keyboard	TK	CI,SA,05
Teletype Printer	TY	CO,LO,06
Teletype Paper Tape Reader	TR	None
Teletype Paper Tape Punch	TP	None
Centronics Line Printer	LP	LO (DOS10E)
Floppy Disk	F0	SF
Floppy Disk	F1	SS,SI,BI,BO,LI,SO

The standard OS Root for LSI-2/20 or LSI-2/60 (DOS20) provides for the following peripherals and default assignments:

Device	Physical Unit	Default Logical Unit
Teletype Keyboard	TK	CI,SA,05
Teletype Printer	TY	CO,06
Teletype Paper Tape Reader	TR	None
Teletype Paper Tape Punch	TP	None
Centronics Line Printer	LP	LO
Floppy Disk Unit 0	F0	SI,BI,LI
Floppy Disk Unit 1	F1	BO,SO
43-Series Disk, Fixed Platter	D0	SF
43-Series Disk, Removable Platter	D1	SS

If either configuration is acceptable, simply load (or autoload) and execute the delivered tape containing the DOS Complete System (ID 96530-41), which already includes the standard OS Root. Continue system generation procedures with Operation of OS:GEN.

The standard OS Root may not be acceptable because support is needed for more, or different, peripherals -- for example, a Data Products printer, magnetic tapes, cassettes, more 43-Series disks, or Double Density disks.

The OS Root routine also supports Distributed Input/Output (DIO) for some peripheral devices, via a standard I/O Distributor or a DMA I/O Distributor. The "Device Selection Table" shows which peripherals are supported by DIO; they include HSPT Reader DIO (PTRD), HSPT Punch DIO (PTPD), Line Printer (LPCED or LPDPD), Card Reader (CRDD), and CRT (CRTD).

Find the page headed "Device Selection Table" in the delivered source listing for OS Root. Determine what "SET YES" statements should be inserted, and perform the modifications, using OMEGA (or load a previous OS and use OS:SFE). Assemble a new version of OS Root. Continue system generation procedures with System Construction.

If 25 IPS magnetic tape drives are to be used with OS, a special modification must be made to OS Root, in addition to insertion of a "SET YES" for each tape unit. Either delete the statement which reads:

```
MACH    SET     0
```

or insert after it a statement which reads:

```
MACH    SET     1
```

Some changes may be required to the default logical unit assignments -- that is, the assignments of logical units to physical devices automatically established for each new /JOB unless overridden by /ASSIGN commands. This may mean modification and re-assembly of OS Root.

The Logical Unit Assignment Table (LUT) source statements contain a DATA statement for each logical unit. For example, the default assignment of SI=PR is fixed by the third operand of the statement:

```
SI      DATA    'SI',PR,PR
```

The second operand is a dummy representing a temporary assignment within one /JOB. To make a default assignment of SI=CR, which is the usual practice for an installation with a card reader, change the statement to:

```
SI      DATA    'SI',CR,CR
```

If the only modifications needed to the standard OS Root are changes to the Logical Unit Assignment Table, the table may be patched in memory (using DEBUG loaded no lower than :2000, or the processor console) before executing OS:GEN. Load (or autoload) the DOS Complete System Tape, but do not execute it. Do the patches, then begin execution at the location labelled OS:GEN on the delivered OS Root load map. Proceed with Operation of OS:GEN.

6.4.2 System Construction

If a new version of the OS Root object tape was generated, it must be combined with the delivered OS Nucleus (ID 9653X-3X) to produce a complete Operating System in low memory.

OS:LNK may be used if a previous OS is available. The OS Root tape must be input from BI before the OS Nucleus tape is input from LI. The output must be another paper tape, which will be the equivalent of the delivered standard DOS Complete System tape, and may be loaded (or autoloaded) and executed to enter OS:GEN. Ready the OS Root tape and enter:

```
/JOB  LINK NEW OS
/ASSIGN  BI=PR,LI=PR
/ASSIGN  BO=PP,LO=LP
/EXEC  OS:LNK,NH,AB=0
OS:LNK  (nn)                    where (nn) indicates the version of OS:LNK
?  LL,TE
OS:LNK  END
```

If no previous OS is available, use the LAMBDA object loader at :2AE0 or higher. Set A, X, and SENSE to zero. Set the Sense Register to :0 for a load map on the teletype, or to :1 for the line printer. OS Root must be input before OS Nucleus. If the final load map shows no unresolved externals, hit RUN to enter OS:GEN.

6.4.3 Operation of OS:GEN

At this point, the resident Operating System has been loaded into memory. It extends from location :0000 to approximately :1DFF, the highest portion of which is a temporary block of code with the entry label OS:GEN.

The system generation procedure has resulted in a transfer of control to OS:GEN. The Operating System about to be recorded for future use will not begin execution at OS:GEN after normal loading (unless OS is loaded from a paper tape), and the locations occupied by OS:GEN will become part of the User Program Area of memory.

OS:GEN converses with the operator thru the operator console. Terminate each response with a Carriage Return. If the message "I/O ERROR" is displayed, the program will execute a processor halt. Hit RUN to restart generation.

The first normal display is:

```
*  ALPHA LSI OS:GEN  *
GEN OS (Y OR N)?
```

A response of "N" will terminate OS:GEN, and control will be passed to OS as if a routine load of OS into memory had just taken place.

A response of "Y" will result in:

```
VOLUME TYPE, UNIT=
```

OS:GEN wants the physical unit on which the Operating System is to be recorded for future use. Typically, the response will be "D0" but any appropriate output device may be named.

NOTE

When floppy disk is used as the output device "F0" should be used. "F1" thru "F3" are valid; but, will produce a system that will not be autoloadable.

After OS has been recorded, the program will ask:

```
GEN AGAIN (Y OR N)?
```

Another copy of OS may now be requested. It is advisable to generate a copy of OS on paper tape as an ultimate back-up. When OS is loaded from paper tape, execution will begin with OS:GEN itself. Thus, once the required configuration of OS has been generated on paper tape, it will never be necessary to go all the way back to the original Root and Nucleus tapes, even if the ordinary system residence file is destroyed.

A response of "N" will pass control to normal OS after the display:

```
*  OS:GEN COMPLETED  *
```

A dummy time and date will be displayed; the operator should supply the actual time and date with /TIME and /DATE commands.

6.4.4 Labelling of System Residence Volume

The Operating System has now been recorded for future use on a disk, a cassette, or a magnetic tape. Before this recording of OS, called the System Residence Volume, can actually be used -- for example, before the System Utility Programs can be copied onto it -- the volume must have an OS volume label. If any one of the following conditions apply, OS:LBL must be run now:

-- The volume has never been labelled for use under OS.
-- The volume has never been used previously as a System Residence Volume.
-- The volume is a reel of magnetic tape.

To label the System Residence Volume, ready the delivered paper tape of OS:LBL, and enter:

```
/ASSIGN  SF=PR
/EXEC  X
```

OS:LBL will begin execution. A typical dialog between the program and the operator would be:

```
NAME
?SYSRES
TYPE AND UNIT
?D0
DOES D0 CONTAIN OS
?Y
SAVE OS
?Y
```

Refer to Section 3.5, OS:LBL, for the remainder of the labelling procedure.

6.5 ADDING SYSTEM UTILITY PROGRAMS

6.5.1 General Considerations

The delivered System Utility Programs should now be added to the same physical volume upon which OS itself resides. First, OS:CPY is loaded and executed from paper tape, and used to copy the same tape to the normal SF device. Then OS:CPY may be executed from SF to copy most of the remaining utilities, including the Link Editor. Finally, certain programs are linked onto SF.

Some thought should be given to the order in which programs are arranged on SF, especially if SF is a magnetic tape or cassette. The most-heavily used programs should be as close to the beginning of the file as possible, to minimize access time. Each installation must decide how to approach this question, or even whether to consider it at all.

6.5.2 Copying OS:CPY

In the examples which follow, the default assignments shown earlier in this chapter are presumed to be re-established at the start of each new job. Recalling that execution of a program from paper tape ignores the program name supplied on the /EXEC command, the following procedure may be used to transfer OS:CPY from the delivered paper tape to the disk containing OS itself.

```
/JOB  COPY OS:CPY TO OS VOLUME
/COMMENT  READY OS:CPY TAPE ON PR
/ASSIGN  SF=PR
/EXEC  X
?
```

OS:CPY has begun execution, and wants parameters. Ready the OS:CPY tape on PR once more, and respond:

```
CB,PR,DO.OS:CPY,TE
```

When the program terminates, OS:CPY is on the disk, from which it may be executed more conveniently for the following steps.

6.5.3 Copying Other Utilities

Section 6.3 identifies the System Utility Programs which may now be copied from the delivered paper tape onto SF. Be careful to ready the proper tape for each "CB" command, so each program gets the right name. In theory, the name given to each System Utility Program on SF need not be the same name used in CAI-supplied documentation. However, those utilities which are loaded into the Transient Area of memory must use the standard names. They are:

OS:LDR
OS:DMP
OS:DBG

The following example shows how OS:LDR and OS:LNK are copied. Ready the OS:LDR tape and enter:

```
/JOB  COPY UTILITIES
/EXEC  OS:CPY
?CB,PR,D0.OS:LDR
?
```

Now ready the OS:LNK tape and respond:

```
CB,PR,D0.OS:LNK
```

Each time OS:CPY finishes with a paper tape, ready another one and give another command:

```
CB,PR,D0.OS:xxx
```

until all the utilities are copied. Then terminate OS:CPY and list the program names for future reference:

```
?TE
/EXEC  OS:VEW,D0,N,TE
```

6.5.4 Linking Utilities

The Operating System Assemblers, and certain System Utilities, must be linked (rather than copied) onto SF from the distributed paper tapes. These programs are identified in Section 6.3, Delivered Software. No special Link Editor parameters are needed:

```
/JOB LINK A UTILITY
/ASSIGN BI=PR,BO=D0.xxxxxx
/EXEC OS:LNK,TE
```

This concludes all system generation procedures, and routine jobs may now be entered.

Appendix A

OS COMMAND SUMMARY

/ASsign	logical unit=physical device (or) logical unit
/BAtch	physical device
/BEgin	[address][,parameters]
/CAncel	
/COmment	[text]
/DAte	[aa/bb/cc]
/EXec	program[,parameters]
/JOb	
/LIst	[logical unit]
/LOad	program
/NJob	
/REsume	[parameters]
/STatus	
/TIme	[hh:mm:ss]
/TYpe	

Appendix B

INPUT/OUTPUT AND EXECUTIVE SERVICES SUMMARY

REQUESTS FOR INPUT/OUTPUT AND RELATED SERVICES

```
LABEL   JST     OPEN:           Open a file.
        DATA    FCB
        JMP     BUSY/ERROR

LABEL   JST     CLOSE:          Close a file.
        DATA    FCB
        JMP     BUSY/ERROR

LABEL   JST     IO:             Perform I/O.
        DATA    IOB
        JMP     BUSY/ERROR

LABEL   JST     WAIT:           Wait for completion of I/O.
        DATA    FCB
        JMP     BUSY/ERROR

LABEL   JST     TEST:           Obtain status of a file.
        DATA    FCB
        JMP     BUSY/ERROR
```

REQUEST FOR EXECUTIVE SERVICES

```
LABEL   JST     SUPV:           Request executive service.
        DATA    SRB

LABEL   JST     MSG:            Print message on CO unit.
        DATA    MSG

LABEL   JST     SPND:           Suspend program execution.
        DATA    MSG

LABEL   JST     TERM:           Terminate program.
```

Appendix C

OS PHYSICAL DEVICE NAMES

TK	Teletype Keyboard
TY	Teletype Printer
TR	Teletype Paper Tape Reader
TP	Teletype Paper Tape Punch
PR	High Speed Paper Tape Reader
PP	High Speed Paper Tape Punch
LP	Line Printer
CR	Card Reader
D0 to Dn	Moving Head Disk Unit 0 to Unit n
M0 to Mn	Open Reel Magnetic Tape Unit 0 to Unit n
C0 to Cn	Digital Cassette Unit 0 to Unit n
F0 to Fn	Floppy Disk Unit 0 to Unit n

Appendix D

OS LOGICAL UNIT NAMES AND STANDARD FUNCTIONS

The function shown for each Logical Unit Name indicates how it is used by CAI-supplied software programs. A user program may use a Logical Unit Name in any FCB for any purpose, although consistency with the standard functions is recommended, and interference with the standard use of SF, CI, and CO is particularly inadvisable.

SF	System File	Any program brought into memory by a /LOAD or /EXEC command, or by a SUPV: request, must be on the current SF.
CI	Command Input	Each system command or additional parameter is entered thru CI. Conversational programs read from CI.
CO	Command Output	Messages from OS are directed to CO. Certain Executive requests from programs produce a message on CO. Conversational programs write to CO.
LO	List Output	Information formatted into lines and pages is written to LO for immediate or deferred printing.
BI	Binary Input	Programs which process object code, such as OS:LDR and OS:LNK, read primary input from BI.
LI	Library Input	Programs which process object code read secondary input from LI.
BO	Binary Output	Programs which generate object code, such as language processors and OS:LNK, write to BO.
SI	Source Input	Input to language processors and to certain file-processing utilities, such as OS:SFE, OS:CNC, and OS:VEW, is read from SI.
SO	Source Output	Programs which generate new source code, such as OS:SFE, write to SO.
SS	Source Save	Language processors store information on SS for a second pass.
SA	Source Alternate	Programs which process source code, such as OS:SFE, read secondary input from SA.
S1 to S4	Scratch 1 Scratch 4	General use. OS:CPY always reads from S1, and writes to S2, after forcing assignments to those units within OS. These assignments still exist after OS:CPY termination, so S2 can be used immediately.
01 to nn	Data Set 01 Data Set nn	General use. 01 to 06 are supplied in the delivered OS, but more may be added during system generation, to a maximum of 99.

Appendix E

OS STANDARD INTERRUPT BLOCKS AND DEVICE ADDRESSES

The interrupt blocks shown here are part of the OS Root Program, as described in the section on System Generation.

INTERRUPT BLOCK	STARTING LOCATION	ENDING LOCATION	DEVICE ADDRESS
Power Up	:0000	:0001	---
Half-Duplex Teletype	:0002	:0007	:07
Moving Head Disk	:000A	:000B	:0F
Real Time Clock	:0018	:001B	:08
Power Down	:001C	:001D	---
Console Interrupt or TRAP	:001E	:001F	---
Open Reel Magnetic Tape	:0022	:0027	:09
HS Paper Tape Reader/Punch	:002A	:002F	:06
Line Printer	:0042	:0047	:04
Card Reader	:004A	:004F	:05
Digital Cassette	:0052	:0057	:10
Floppy Disk	-----	-----	:11

Appendix F

PERIPHERAL DEVICES SUPPORTED UNDER OS

DEVICE	MAKE and MODEL	INTERFACE REQUIRED
HSPT Reader	Remex	53223
	Digitronix	53223
HSPT Punch	Remex	53223
	Facit	53223
Card Reader	Bridge 8000	53223
	Peripheral Dynamics C301	53223
	Documation	53223
Line Printer	Centronics 101	53223
	Data Products 2310	53223
	Data Products 2410	53223
Magnetic Tape	Pertec 7820	53224-Y7 or above OR 53224-Z1 or above
Digital Cassette	Computer Automation, Inc.	53240-X4 or above.
Moving Head Disk	Diablo Model 31/33,41/43	53264-Y2 or above AND 53263-Y0 or above
Floppy Disk	Computer Automation, Inc.	53566

```
PAGE  0001  11/05/74  17:26:05  ALPHA 16/LSI DISK LOADER

LINE  LOC  INST ADDR  LABEL  MNEM OPERAND  COMMENT
0002                  *
0003                  *           96537-A0
0004                  *
0005                  *COPYRIGHT 1974 COMPUTER AUTOMATION INC
0006                  *
0007  2FD0                      REL  :2FD0     BINARY RELOCATIBLE
0008        000F      DA        EQU  017       DISC DEVICE ADDRESS
0009                  *
0010  2FD0 FA00 2FD1  START     JST  $+1       WHERE AM I?
0011  2FD1 0000                 DATA 0
0012  2FD2 E601 2FD1  LOC       LDX  $-1
0013  2FD3 C20C                 AXI  DELTA     POINT TO TABLE
0014  2FD4 6F78       AGAIN     WRX  DA,0      READ FROM DISC
0015  2FD5 5978                 RDA  DA,0      GET STATUS
0016  2FD6 1354                 LLA  5
0017  2FD7 2104 2FDC            JAZ  $+5       OK?
0018  2FD8 407C                 SEL  DA,4      NO
0019  2FD9 497C                 SEN  DA,4      RESET AND WAIT
0020  2FDA F601 2FD9            JMP  $-1
0021  2FDB F607 2FD4            JMP  AGAIN     TRY AGAIN
0022  2FDC F000 0000            JMP  0         GOTO BOOT #2
0023        000C      DELTA     EQU  $-LOC+1
0024  2FDD 0000                 DATA 0,4,256,0  DISC CONTROL WORDS
      2FDE 0004
      2FDF 0100
      2FE0 0000
0025        2FD0                END  START

0000  ERRORS
```

Appendix G

BOOTSTRAP FOR DOS WITHOUT AUTOLOAD

```
PAGE  0001  11/01/74  02:18:48  ALPHA 16/LSI OS MAG TAPE LOADER

LINE  LOC  INST ADDR  LABEL   MNEM OPERAND  COMMENT
0002                  *
0003                  *            96535-D0
0004                  *
0005                  *COPYRIGHT 1974 COMPUTER AUTOMATION INC
0006                  *
0007  2800            .        REL  :2800    LOCATE BOOT @ OR > :2800
0008        0009      MT       EQU  9        MT DEVICE ADDRESS
0009        2800      TBOOT    EQU  $
0010  2800 404C       TBOOT2   SEL  MT,4     ERROR, RE-INITIALIZE
0011  2801 404A                SEL  MT,2     REWIND TAPE
0012  2802 494B                SEN  MT,3     WAIT UNTIL DONE
0013  2803 F601 2802           JMP  $-1
0014  2804 0108                ZXR           STARTING LOAD LOCATION @ 0
0015  2805 494B                SEN  MT,3     WAIT UNTIL TAPE READY
0016  2806 F601 2805           JMP  $-1
0017  2807 404C                SEL  MT,4     INITIALIZE TAPE
0018  2808 404F                SEL  MT,7     SELECT READ MODE
0019  2809 5949       TBOOT1   RDA  MT,1     READ HIGH ORDER BYTE
0020  280A 1357                LLA  8        MOVE IT OVER
0021  280B 484D                SSN  MT,5     END OF RECORD?
0022  280C F60C 2800           JMP  TBOOT2   YES, LOST A BYTE
0023  280D 4949                SEN  MT,1     BUFFER READY?
0024  280E F603 2808           JMP  $-3      NO, KEEP TESTING
0025  280F 7949                RBA  MT,1     READ LOW ORDER BYTE
0026  2810 484D                SSN  MT,5     END OF RECORD?
0027  2811 F205 2817           JMP  TBOOT3   YES, DONT STORE CRC
0028  2812 4949                SEN  MT,1     BUFFER READY?
0029  2813 F603 2810           JMP  $-3      NO, KEEP TESTING
0030  2814 9C00 0000           STA  @0       YES, STORE WORD
0031  2815 0128                IXR           INCREMENT ADDRESS POINTER
0032  2816 F60D 2809           JMP  TBOOT1   READ NEXT WORD
0033  2817 484C       TBOOT3   SSN  MT,4     PARITY ERROR?
0034  2818 F618 2800           JMP  TBOOT2   YES, DO IT OVER
0035  2819 404C                SEL  MT,4     NO, INITIALIZE TAPE
0036  281A F000 0000           JMP  0        BEGIN EXECUTION AT LOC :0000
0037        2800               END  TBOOT

0000  ERRORS
```

Appendix H

BOOTSTRAP FOR MTOS WITHOUT AUTOLOAD

```
PAGE  0001  11/01/74  02:19:01  ALPHA 16/LSI CASSETTE LOADER

LINE  LOC   INST ADDR  LABEL   MNEM OPERAND  COMMENT
0002                  *
0003                  *           96536-00
0004                  *
0005                  *COPYRIGHT 1974       COMPUTER AUTOMATION INC
0006                  *
0007  2800                      REL  :2800    LOCATE BOOT = OR > :2800
0008        0010      CA        EQU  16       CASSETTE DEVICE ADDRESS
0009        2800      CBOOT     EQU  $
0010  2800  4084      CBOOT3    SEL  CA,4     ERROR HANDLING, INITIALIZE
0011  2801  C60A                LAP  :A       ISSUE REWIND
0012  2802  6D83                WRA  CA,3
0013  2803  0100                ZXR           STARTING LOAD LOC = 0
0014  2804  4084                SEL  CA,4     INITIALIZE CASSETTE
0015  2805  6983      CBOOT4    WRZ  CA,3     ISSUE READ MODE FUNCTION
0016  2806  4884      CBOOT1    SSN  CA,4     END OF RECORD?
0017  2807  F20C 2814           JMP  CBOOT2   YES
0018  2808  4987                SEN  CA,7     BUFFER READY?
0019  2809  F603 2806           JMP  CBOOT1   NO, KEEP CHECKING
0020  280A  5987                RDA  CA,7     READ HIGH ORDER BYTE
0021  280B  1357                LLA  8        MOVE IT OVER
0022  280C  4884                SSN  CA,4     END OF RECORD?
0023  280D  F60D 2800           JMP  CBOOT3   YES, LOST A BYTE
0024  280E  4987                SEN  CA,7     BUFFER READY?
0025  280F  F603 280C           JMP  $-3      NO, KEEP TESTING
0026  2810  7987                RBA  CA,7     READ LOW ORDER BYTE
0027  2811  9C00 0000           STA  *0       STORE A WORD
0028  2812  0128                IXR           INCREMENT ADDRESS POINTER
0029  2813  F60D 2806           JMP  CBOOT1   READ NEXT WORD
0030  2814  C610      CBOOT2    LAP  :10      TEST FOR PARITY ERROR
0031  2815  5D82                RDAM CA,2     0
0032  2816  3156 2800           JAN  CBOOT3   YES,PROCESS ERROR
0033  2817  4084                SEL  CA,4     INITIALIZE
0034  2818  F000 0000           JMP  0        EXECUTE AT LOC 0
0035        2800                END  CBOOT

0000  ERRORS
```

Appendix I

BOOTSTRAP FOR COS WITHOUT AUTOLOAD

ComputerAutomation
18651 Von Karman, Irvine, Calif.

ENGINEERING NOTICE

NO. R1

5 6 6 1 S

DOCUMENT NO.	REV. IS	REV. WAS	TITLE	INCORP DATE
96530	D7	D7	SOFT - OPERATING SYSTEM	

CLASS

- A-MAND/FUNC [X]
- B-NON-MAND/FUNC [ ]
- C-RECORD CHG [ ]
- D-DEVIATION [ ]
- STOP ORDER [ ]
- AEN [ ]

- HARDWARE CHG. [ ]
- SOFTWARE CHG. [X]
- PUBLICATIONS CHG. [ ]
- CAPABLE CHG. [ ]

EFFECTIVITY:

REASON FOR CHANGE:

REA NO. 02450S

CO-ORD WITH:

If a disk is physically WRITE-PROTECTED and has only one partition, OS will declare a WRITE-PROTECT error correctly. However, if the user clears the physical WRITE-PROTECT but does not do a /JOB before the second OPEN-for-WRITE request, then the WRITE PROTECT error will again be reported.

DISPOSITION

ACTIVITY	U	R	S
ON ORDER			
IN STOCK			
IPP			
IPT			
FIN GOODS			
CUST RET			
NOTIFY VEND			

AFFECTED ITEMS:

- SOFTWARE PROG.
- PUBLICATIONS
- CAP. PROGRAMS
- CONFIGURATIONS
- PROCEDURES
- TOOLING
- TEST EQUIP.

DESCRIPTION OF CHANGE:

A. 96530-00 SOFT - OPERATING SYSTEM

Do a /JOB command after a physical WRITE-PROTECTED disk has been cleared for writing.

APPROVALS

ENGR.	—
SOFTWARE	MS
Q.C.	
CAP. TEST	
PROD. CONT.	
MATERIALS	—
TEST ENGR.	
TECH SERV	
CUST SERV	
IND ENG	
PUBLICATIONS	—

DRAWN BY: J. FINE
DATE: 4/12/77
CHKD BY:
REL BY:
DATE: 4/14/77

NOTE ON ISSUE D7 OF OS

The version of the Operating System delivered with this package is D7. (The D6 revision level OS was a documentation change which is incorporated in the updated User Manual sheets.)

The new OS comprises:

OS ROOT	96529-20D7
OS NUCLEUS	96530-30D7

Apart from correcting all outstanding errors, the main feature of the new version is the inclusion of DIOS device drivers for the following device-types:

Line Printer (Data Products or Centronics)
High Speed Paper Tape Reader
High Speed Paper Tape Punch
Card Reader
CRT

N. B. The CRT is assumed to be on DIOS channel 2

It is intended that OS will be configured by you either for DIOS peripherals or for non - DIOS peripherals, not a combination, and only 1 unit of each device-type is allowed. However, if you do configure with mixed DIOS and non-DIOS devices, take care when editing ROOT, to ensure that LOBP: (Page 20 of the ROOT Listing) is set correctly - deleting lines 571 - 573 will be best.
If a CRT is included in your configuration, it will not replace the teletype connected to the Option Card. Instead it will be used as device TV both for input and output.

Finally on this new OS, please note the instructions on Page 1 of the ROOT Listing concerning the jumpering of the DIOS to offset interrupts to: 100. This must be done.
Included with the new OS software is an updated version of OS:HDR for use with a DIOS - connected HSPTP. NOTE: Do not use this new OS:HDR with an old OS. Operating instructions are still the same.

C.A.I. Limited
EUROPEAN TECHNICAL SUPPORT GROUP

December 1976

Note on OS: LDR

OS: LDR 96542-C0

When a program which writes to scratchpad locations is loaded by OS: LDR, the OS user scratchpad usage information is not correctly updated.

The following observations should be noted:-

1. Information returned after the use of the OS Status command or executive service SUPV Code 1 may not be accurate.

2. The loaded program should, however, be able to correctly access the required scratchpad locations.

3. Scratchpad violation by overlayed programs may not be detected.

C.A.I. Limited

EUROPEAN TECHNICAL OPERATIONS

March, 1977

NOTE ON OS - DELETED FILES

O.S. 96530 - D7

For those users who find themselves in the position where they need to recover information from a file which has been accidentally or prematurely deleted, it is possible to apply a temporary patch to the File Manager which will enable OS:CPY to copy the "deleted" file to a new active file.

For this patch to work successfully, there must not be any other previously deleted file (on the same disk) bearing the same name as the one you wish to recover.

The patch is as follows:-

Location	Old Contents	New Contents
FM:OPN + :127	:2081	:F83B
:003C	:083C	:1150
:003D	:083D	:13D0
:003E	:083E	:9C09
:003F	:083F	:F704

This patch may be applied to using OS:DBG, and then OS:CPY can be used in the normal way. E.g. if a source file BASE on D1 has been deleted, then, after patching, do:

```
/EX  OS:CPY,CS,D1.BASE,D1.BASE2,TE
```

Having accomplished the recovery, OS should be re-loaded.

CAI Limited
EUROPEAN TECHNICAL SUPPORT GROUP
EUROPEAN TECHNICAL OPERATIONS

March 1977

SOFTWARE ERRATA NOTICE

PROGRAM NAME	PROGRAM ID	ERRATA #	DATE
OS ROOT	96529-D7	823	12/13/76

DESCRIPTION OF PROBLEM

A Data Products Line Printer will not line-feed properly when Root is set up to assume a Data Products printer.

EFFECTIVITY (VERSION)

Version D7 of OS Root, applicable only to OS usage with Data Products line printers.

DESCRIPTION OF CHANGE

The source file of OS Root should be edited and reassembled so that the value in LP:UCB+4 becomes :0201, rather than :0200. In the standard supplied listing of OS Root, this change is required at line no. 892. Alternatively, OS may be patched in memory at location LP:UCB+4 (determined from link map) as follows:

LOCATION	OLD CONTENTS	NEW CONTENTS
LP:UCB+4 (LD:UCB+4 if DIO)	:0200	:0201

APPROVED BY:

Note on OS Overlay Facilities

OS NUCLEUS 96530-D7

The OS Executive service call SUPV: code 8 requests the loading of a program overlay.

In the OS Manual, attention is drawn to the fact that the requested load must fall completely within the User Area of Memory and must not overlay the request coding sequence nor related SRB. The OS will trap violation of User Area of Memory and scratchpad locations used by the request program. It is recommended that the User ensures that an overlay program will not overlay memory locations unprotected by the OS.

C. A. I. Limited
EUROPEAN TECHNICAL OPERATIONS

March, 1977

SOFTWARE ERRATA NOTICE

PROGRAM NAME	PROGRAM ID	ERRATA #	DATE
OS FILE MANAGER	96533-D1	243	10/22/76

DESCRIPTION OF PROBLEM

It is possible to create two files with the same name on a disk, if the second is opened before the first has closed.

EFFECTIVITY (VERSION)

Applies to OS Version 96530-D7 and all previous versions.

DESCRIPTION OF CHANGE

This situation can only occur if two different LUNs are assigned to the same file name on the same physical unit in an attempt to create two new files. Users should avoid this situation when making file assignments.

APPROVED BY:

DOR

ComputerAutomation

SOFTWARE ERRATA NOTICE

PROGRAM NAME	PROGRAM ID	ERRATA #	DATE
OS:CNC	96548-A1	494	9/1/76

DESCRIPTION OF PROBLEM

Batch mode causes resetting of source line numbers between each concordance. The current line number should be saved, so that the numbers will match the corresponding assembly listings.

EFFECTIVITY (VERSION)

Version 96548-A1

DESCRIPTION OF CHANGE

After loading OS:CNC, but prior to execution, make the following patches with OS:DBG:

Location	Old Contents	New Contents
:80	--	:0110
:81	--	:F182
:82	--	:0061R0
:005DR0	:F201	:F080

APPROVED BY:

ComputerAutomation

SOFTWARE ERRATA NOTICE

PROGRAM NAME	PROGRAM ID	ERRATA #	DATE
OS:CNC	96548-A1	611	9/1/76

DESCRIPTION OF PROBLEM

OS:CNC does not list references to labels used as operands in the following instructions:

DVD
DVS
MPS
MPY
NRM

EFFECTIVITY (VERSION)

Version 96548-A1

DESCRIPTION OF CHANGE

After loading OS:CNC, but prior to execution, make the following patches with OS:DBG:

Location	Old Contents	New Contents
:0525R0	:D6C4	:C9CE
:0526R0	:C4D6	:C4C9
:0527R0	:D3C4	:CEC4
:0543R0	:D0D3	:D0C5
:0545R0	:D9CE	:C5CE
:054ER0	:CDCE	:C1CE

APPROVED BY:

ComputerAutomation

SOFTWARE ERRATA NOTICE

PROGRAM NAME	PROGRAM ID	ERRATA #	DATE
FSAVE	96955-A2	747	8/25/76

DESCRIPTION OF PROBLEM

FSAVE will copy 43-type disk drives (200 cylinders), but not 44-type (400 cylinders). The patch below will allow copying of any size drive up to 410 cylinders.

EFFECTIVITY (VERSION)

Version 96955-A2

DESCRIPTION OF CHANGE

After loading FSAVE into memory, but prior to execution, make the following patches:

LOCATION	OLD CONTENTS	NEW CONTENTS
:1D7F	:12C0	:2670
:1D8F	:12C0	:2670
:2068	:12C0	:2670
:2078	:12C0	:2670

APPROVED BY:

Note on Scratchpad Map and OS:LNK

OS:LNK 96549 - B2

For users with a Teletype as the only listing device, it is possible to modify OS:LNK to cause the Scratchpad Usage Table to be omitted from the listing, thereby reducing the time taken to produce the link-load details:

Location	Old Content	New Content
OS:LNK + :A66	:FA92	:F22F

C.A.I. Limited
EUROPEAN TECHNICAL SUPPORT GROUP

December, 1976.

Preliminary Errata Notice for OS:ILD

OS:ILD 96551 - A2

Magnetic Tape users only

It is not possible to load a program from M1, M2 or M3. If it is attempted, OS:ILD loads the first record correctly, then, after relocating itself to upper memory, it will load the next record from MØ.

In order to allow loading from other than MØ, users should modify OS:ILD as follows:

Location	Old Contents	New Contents
OS:ILD + :1Ø6	:F235	:C28Ø
+ :1ØB	:C6Ø3	:ØØ48
C	:82ØC	:C6Ø3
D	:9BØC	:82ØB
E	:C7Ø4	:C3CD
F	:8BØA	:38Ø5
:11Ø	:C4Ø8	:1128
1	:13A8	:ØØDØ
2	:Ø15Ø	:3ØC2
3	:3142	:EBØC
4	:1328	:F229
5	:EBØA	:9BØ4

C.A.I. Limited
EUROPEAN TECHNICAL SUPPORT GROUP

December, 1976.

ComputerAutomation

SOFTWARE ERRATA NOTICE

PROGRAM NAME	PROGRAM ID	ERRATA #	DATE
OS:ILD	96551-A2	634	6/22/76

DESCRIPTION OF PROBLEM

Program loading from Floppy Disk is handled incorrectly when the program to be loaded falls across a partition boundary.

EFFECTIVITY (VERSION)

96551-A2

DESCRIPTION OF CHANGE

Load OS:ILD, then make the following patches with OS:DBG prior to execution:

LOCATION	OLD CONTENTS	NEW CONTENTS
:3FER0	:E217	:E20F
:40BR0	:0000	:0050

APPROVED BY:

Preliminary Errata Notice for OS:VEW

OS:VEW 96545 - C2

The F command, which lists the contents of a specified file, produces a 78 - character output line. This causes the last 6 characters to overprint each other when output is directed to the Teletype printer.

For users with a Teletype as the only listing device, the following patch will cause only 72 characters to be printed per line, the first 6 blanks (spaces) being omitted from the print buffer:

Location	Old Contents	New Contents
OS:VEW+:555	:C64E	:C648
+:557	:FF79	:C2Ø3
8	:B659	:FF7A
9	:21Ø1	:B65A
A	:F63A	:317A

The Patch can be done using OS:DBG and then a new file may be created using OS:DMP.
Note: the new program file is only valid for the OS under which the modification was made. If OS is reconfigured, OS:VEW must be re-modified, and the old modified file must be deleted.

C.A.I. Limited
EUROPEAN TECHNICAL SUPPORT GROUP

December, 1976.

ERRATA NOTICE FOR OS:ILD FOR USE ON FLOPPY DISC

OS:ILD - 96551-A2

OS:ILD sometimes has difficulties loading files from Floppy Diskettes due to incorrect handling of records which cross partition boundaries.
The following patch cures this fault.
In addition, a second 1-word patch ensures a correct first-time seek.

Address	Old Contents	New Contents
OS:ILD + :3FE	:E217	:E20F
OS:ILD + :40B	:0000	:0050

Patch implementation:

The following method should be used to carry out the above patching.
Note that the new file is absolute and should be re-created whenever OS is re-configured:- (user typing is underlined).

```
> /LOAD  OS:ILD
> /EX    OS:DBG
   OS:DBG   3BF7
? I3FER0
1BC0  E217  E20F
? I40BR0
1BCD  0000  0050
? T
> /AS  BO = F1.NEWILD
> /EX  OS:DMP
```

C.A.I. Limited
EUROPEAN TECHNICAL SUPPORT GROUP

July 1976.

ComputerAutomation

SOFTWARE ERRATA NOTICE

PROGRAM NAME	PROGRAM ID	ERRATA #	DATE
OS:DBG	96546-A1	464	6/24/75

DESCRIPTION OF PROBLEM

A CI command file which calls OS:DBG (/EX OS:DBG, etc.) will not be executed correctly from a file type device (disk, floppy disk, magnetic tape) since OS:DBG uses OPEN: and CLOSE: calls to the CI unit internally, rather than using SUPV: calls. This effectively causes a rewind of the CI file when it is first OPENed by OS:DBG.

EFFECTIVITY (VERSION)

Version A1

DESCRIPTION OF CHANGE

OS:DBG may be included in /BAtch job control sequences from non-bulk devices only.

APPROVED BY: