pdp11

IAS User's Guide

Order No. DEC-11-OIUGA-C-D

digital

IAS
User's Guide

Order No. DEC-11-OIUGA-C-D

digital equipment corporation · maynard, massachusetts

First Printing, December 1975
Revised: May 1976
Revised: September 1977

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

CHAPTER 1

INTRODUCTION TO IAS

1.1 IAS

IAS (Interactive Applications System) is a multifunction operating system for the PDP-11 computers. IAS supports the concurrent execution of three processing modes: Interactive, Batch and Real-time. Real-time operates on a priority basis, while Interactive and Batch are timeshared. All three processing modes are made available through commands issued at a user's terminal. Final control of the system's resources is retained by the system manager.

This manual provides a guide to the user's terminal interface with the system.

Every user identifies himself to the system by supplying a unique User Name when he logs in at a terminal. The user must also supply a password that has been previously associated with the User Name.

Also associated with each user name are User Identification Code (UIC) and the user's privilege masks. The UIC consists of two parts, which represent a group of users and the user's identity in the group. The system uses the UIC to control access to programs and data within and between groups. With privilege masks the manager can specify that various sets of commands be made available to a particular user.

1.1.1 Real-time Processing

IAS provides the same real-time capabilities as DIGITAL's RSX-11D multiprogramming system. These capabilities are designed for applications that require response to physical events as they occur. Typical real-time applications include manufacturing process control, laboratory data acquisition, and communications.

1.1.2 Interactive Processing

Operating from an interactive terminal, a user may create and run programs interactively or submit them to a batch stream; alternatively, the user may exploit other users' programs or standard programs provided by the system (if his UIC grants him access). Even though many users benefit from sharing programs and files, the system preserves the individual user's privacy and shields the activities of other users.

Interactive processing offers 2-way communication with the computer. The user initiates activities and remains in control but regulates those activities according to information that the system feeds back to him.

The interactive user communicates with the system by typing commands at the keyboard of a terminal. The standard IAS command language for the general user is PDS (Program Development System), which is described in this manual; but the user has the option of creating other command interfaces to suit particular applications.

1.1.3 Batch Processing

A batch job is a collection of commands that, once submitted to the batch stream, will continue to completion without user intervention. Under IAS, programs can be created and tested interactively, and then submitted to a batch stream for execution; or again, creation, testing and execution might all

In batch mode, as in interactive, programs can be compiled or assembled, linked, and executed; devices can be claimed and released, and messages can be sent to the operator. All of these services are invoked by the same commands used for interactive processing. In interactive mode, the user can store these commands in a file which is then submitted to the batch processor. Alternatively, the batch commands may be submitted on punched cards.

Since batch requirements vary from installation to installation, and even from day to day, the IAS batch facility can be readily adjusted to meet the needs of a particular installation. For example, consider a system manager faced with a large number of daytime interactive users and a number of large batch jobs. The system manager could allocate 90% of the system's resources to interactive use during the day, and reverse the allocation at night. This would allow some batch jobs to be run during the day and some interactive jobs at night.

1.2 IAS COMMAND LANGUAGE

The standard IAS interface for all terminal and batch users is provided by the Program Development System (PDS).

1.2.1 PDS Commands

Under PDS, interactive users may create, compile, link and run programs; submit jobs to a batch stream; use various peripheral devices and obtain information about the system.

PDS is a command interpreter. After PDS is activated at a terminal, either automatically or by CTRL/C (Control C), PDS invites the input of a command by issuing the prompt "PDS>". The user must provide identification to the system by logging in (entering User Name and password) before beginning terminal activity. After logging in, the user is able to make use of those IAS facilities allocated to him by the system manager (see Section 1.2.3).

A typical sequence of activities during a terminal session might involve entering a source program, translating it into machine-executable form, and then running the program. The user requires the services of a number of system programs to do these things: an editor to enter the source program and to correct typographical and other errors; a language translator to convert the source program into object code; and, for most programs, the Task Builder to create an executable task.

Commands input to PDS invoke the services of these programs. PDS checks to ensure that input commands are meaningful in the current context. For example, the FORTRAN command may only be issued after a user has logged in.

1.2.1.1 Indirect Commands - Rather than type a set of terminal commands interactively, the user can create a file and enter the commands for future use. Such a file can be edited to allow for corrections or second thoughts. It can also save time with commonly used sequences of commands.

To execute the commands the user types @ followed by the name of the file. In this way the command sequence is still initiated under the control of the user.

1.2.1.2 Batch Commands - Most PDS commands can be used in a batch stream. Batch commands always contain a dollar sign ($) in the first position of the line, e.g. $RUN. The batch user, like the interactive user, can use PDS commands to create, compile, link and run programs, and to use various peripherals.

An interactive user may create a file of batch commands and submit the file to a batch stream; alternatively the batch job could be submitted on cards.

Interactive and batch commands are described in parallel in Part 2. The parameters of an interactive PDS command may be either prompted for, supplied as one line (with continuation characters where necessary), or issued in a combination of both methods; whereas all the parameters of a batch command must be supplied as one string, using continuation characters between lines when necessary. The command descriptions provide examples of both interactive and batch usage.

1.2.3 Restricting the Use of PDS Commands

An IAS user may discover that the system does not allow him to issue certain commands from an interactive terminal or within a batch job. This situation can occur because every manager of an IAS System determines the groups of commands that each user is allowed to issue. See Chapter 4 of this manual.

For instance, the manager may decide that a certain user may only program in BASIC, and therefore allocates that user only the commands necessary for developing and running BASIC programs.

IAS allows the system manager to control the way the command language is used so that the installation's work can be carried out as efficiently as possible.

1.3 PROGRAMMING LANGUAGES

IAS supports several programming languages, including BASIC, COBOL, FORTRAN, CORAL 66 and MACRO-11. The MACRO-11 Assembler is shipped with IAS, whereas translators for the other languages are optional.

BASIC programs can be executed immediately after translation, because they produce intermediate code which is run by an interpreter. FORTRAN, CORAL 66, COBOL and MACRO-11 produce machine-language code and therefore require the additional step of linking.

1.3.1 BASIC

BASIC is easy to learn and use, and has found wide acceptance in educational, business, and scientific applications. BASIC-11's "immediate" mode allows each statement to be executed as it is typed in; the computer can be used like a desk calculator. Alternatively, a program can be entered, edited and then run as a unit.

1.3.2 COBOL

COBOL (COmmon Business Oriented Language) is a pseudo-English programming language designed primarily for business use. PDP-11 COBOL conforms to the American National Standard 1974 level-1 COBOL standard, with many high-level features. COBOL is an optional feature of IAS.

COBOL can be used in both batch and interactive applications. For situations where the terminal is the only input device, PDP-11 COBOL provides a simple, terminal-oriented line format. Several utility programs are provided with COBOL, including a report-generating program and a reformatting program.

1.3.3 FORTRAN

The FORTRAN (FORmula TRANslation) language is especially useful in scientific and mathematical applications. PDP-11 FORTRAN conforms to the specifications of American National Standard FORTRAN (X3.9-1966), with substantial extensions to that standard.

The FORTRAN system consists of a compiler, a library of functions, and an object time system (OTS). The compiler produces object code from the source program. The OTS consists of routines that are selectively linked with the user's program to perform certain arithmetic, I/O, and system-dependent service operations. The OTS also detects and reports run-time error conditions.

There are two FORTRAN compilers supported on IAS: FORTRAN-IV and FORTRAN IV-PLUS.

1.3.4 MACRO-11

The programmer who wishes to work closely with the PDP-11 hardware and IAS may use the powerful MACRO-11 assembler. In addition to allowing the user to invoke machine-language instructions, MACRO-11 allows the programmer to define "macros" which may be invoked to generate repetitive coding sequences. The MACRO-11 language can be used both in interactive and batch processing applications.

1.3.5 CORAL

CORAL 66 (Computer On-line Real-time Applications Language) is a general purpose programming language based on ALGOL 60. with some features from CORAL 64, JOVIAL, and some FORTRAN. It is the standard implemention language for British military and government applications and has gained widespread acceptance in the OEM fields for the implemention of real time and transaction processing systems. The language itself is designed to generate efficient code and, although high-level in concept, it has low-level features (such as embedded machine code) to ensure total flexibilty.

CHAPTER 2

A SAMPLE INTERACTIVE SESSION

This chapter introduces the user to PDS by demonstrating its use in a typical session at an interactive terminal. Section 2.1 records the session, which is then described line by line in the following sections.

The line numbers at the left hand margin of the page are for reference purposes and are not part of the actual session. Underlining indicates text printed by the system.

2.1 SAMPLE SESSION

```
01         IAS  PROGRAM DEVELOPMENT SYSTEM  VERSION 2

                    17:09:08       15-MAY-77

02     PDS> LOGIN/NONOTICE

03     USER NAME? CAROL

04     PASSWORD?

05     USER CAROL  UIC [200,22]  TT05:  JOB-ID 160 17:09:21  15-MAY-77

06     PDS> CREATE ADD.FTN

       READY FOR INPUT

07             TYPE 1

08     1       FORMAT(' ENTER TWO NUMBERS')

09             APPE\EPP^R

10             ACCEPT 2,K,L

11     2       FORMAT (22\2\I5)

12             PRINT^U

13             TYPE 3,K+L

14     3       FORMAT(' THE SUM IS ',I5)

15             STOP
```

```
16            END
17    ^Z
18    PDS> TYPE ADD.FTN
19            TYPE 1
20    1       FORMAT (' ENTER TWO NUMBERS')
21            ACCEPT 2,K,L
22    2       FORMAT (2I5)
23            TYPE 3,K+L
24    3       FORMAT (' THE SUM IS ',I5)
25            STOP
26            END

27    PDS> FORTRAN   ADD
28    17:17:41  SIZE: 10K  CPU: 0.10

29    PDS> LINK   ADD
30    17:18:38  SIZE: 11K  CPU: 12.06  STATUS: SUCCESS

31    PDS> RUN   ADD
32    17:30:51
33    ENTER TWO NUMBERS
34    12,  78
35    THE SUM IS    90
36    JOB160 -- STOP
37    17:31:14  SIZE: 7K  CPU: 0.02
38    PDS> DIRECTORY

39    DIRECTORY DB0:[200,22]

40    15-MAY-77 17:36
41    ADD.OBJ;1           2.             15-MAY-77 17:17
42    ADD.FTN;1           1.             15-MAY-77 17:17
43    ADD.TSK;1           32.      C     15-MAY-77 17:18
44            TOTAL OF 35./35. BLOCKS IN 3. FILES
```

```
45      PDS> RENAME   ADD.*;*    ADDTWO.*;*

46      PDS> DIREXCTORY/BRIEF^U

47      PDS> DIRECTORY/BRIEF

48      DIRECTORY DB0:[200,22]

49      ADDTWO.OBJ;1

50      ADDTWO.FTN;1

51      ADDTWO.TSK;1

52      PDS> LOGOUT

53      USER CAROL UIC [200,22] TT05: JOB-ID 160 17:45:01   15-MAY-77

54      CONNECT TIME 14 M    SYSTEM UTILIZATION 12 MCTS

55      BYE
```

2.2 INVOKING PDS

The Program Development System (PDS) is the standard IAS interface the user has to the computer. The installation's system manager determines who may use PDS and decides which terminals will support it.

Therefore, in order to issue PDS commands at a terminal, a user must be authorized to do so, and the terminal must support PDS. If these two conditions have been satisfied, then the following steps should be taken to invoke PDS:

1. Check that the terminal's power is on.

2. Set the LOCAL/REMOTE switch to REMOTE.

3. Consult installation instructions for additional required terminal settings and dial-up instructions.

4. Press CTRL/C (that is, type C while holding down the CTRL key).

The system responds to CTRL/C by displaying a PDS identifier, the current time and date. For example:

```
                IAS PROGRAM DEVELOPMENT SYSTEM   VERSION 2

                          17:09:08         15-MAY-77

        PDS>
```

The prompt PDS> is then displayed at the beginning of the next line to indicate that the system is ready to receive PDS commands. If a notice is to be printed at log in, this will be displayed before the next PDS> prompt (see Section 2.3.1).

In some instances the user may discover that a terminal is already prompting for PDS commands even though no one else is currently using that terminal. A user can then log into the system immediately since PDS has already been invoked.

PDS is designed to time out after several minutes (the exact number of minutes depends on the installation) if nothing has been typed and no program is running. When this happens, the system displays the messages

```
TIMEOUT

BYE
```

The user must then type CTRL/C to re-activate PDS.

2.3 PDS COMMANDS

2.3.1 The LOGIN Command

Once PDS is prompting, the user initiates an interactive session by typing

```
LOGIN <CR>
```

The symbol <CR> represents carriage return, which may be activated either by the carriage return key (<CR> or RETURN) or by the altmode key (ESC or ALT). One of these keys must be pressed to terminate a command string or any other line of input and to transmit the line to the system. The carriage return key and the altmode key can have different effects in certain contexts. The differences are discussed in Chapter 4, Section 4.1.2.

2.3.1.1 The User Name

In response to LOGIN, PDS displays the prompt

```
USER NAME?
```

which asks the user to supply his User Name. The User Name is a unique 1- to 12-character alphanumeric string that identifies the individual user to the system. The system then finds the user's User Identification Code (UIC) given when the user was authorized. The UIC determines whether the user is allowed to read or manipulate any file he attempts to access. See Chapter 6, Section 6.1.3 for further details.

NOTE

The system manager assigns each user a User Name, which is then registered with IAS. A user who does not have a User Name or has forgotten it should consult the system manager.

2.3.1.2 The Password - An additional security measure to prevent unauthorized access to the system is the user's password. Once the user has entered a User Name by activating carriage return, PDS prompts

```
     PASSWORD?
```

at the beginning of the next line. The user must then type in a 1- to 6-character alphanumeric string, i.e. a password, that has previously been associated with the unique User Name.

A user may change his password with the SET PASSWORD command (see Part 2).

Since the purpose of the password is to verify a user's identity, it should be kept secret. PDS respects the user's private password by not displaying the characters typed in after the PASSWORD? prompt.

If the password given is incorrect, PDS prompts PASSWORD? again. The user has three chances to type the password correctly before PDS exits and prints the text BYE. To begin again, the user must type CTRL/C and then LOGIN. When the user types the correct password, IAS responds by displaying the following information (line 5):

```
USER CAROL   UIC [200,22]   TT05:   JOB-ID 160   17:09:21   15-MAY-77
```

The JOB-ID number is assigned to the session by IAS and is normally significant only to the system manager or operator who oversees the running of the whole computer system.

The above line is followed by a new line beginning with PDS> to indicate that the system is ready to receive further commands.

2.4 THE CREATE COMMAND

After successfully logging in, the user creates a file called ADD.FTN (line 6). The CREATE command is one of several PDS commands that can be used to create a file. "ADD" is the filename and "FTN" is the file type, which describes the contents of the file. In this case, the filetype indicates that the file contains a FORTRAN source program (see Table 6-2 for IAS default filetypes).

After terminating the CREATE command by pressing carriage return, the user starts to enter the source program lines from the keyboard. The first typing position on each line is equivalent to position 1 on a coding sheet or punched card. The various function keys (described in Chapter 3) must be used to format the lines as required. For example, the TAB key may be used to skip 8 spaces to position the text "TYPE 1" in line 7. Carriage return terminates each line and moves the typing position to position 1 of the next line.

2.4.1 Correcting Input Errors

On line 9, the user makes a typing error, corrected by means of the DELETE key (sometimes labelled RUBOUT). The user presses the key three times to delete E, P and then P again. The characters deleted are echoed on the terminal as follows:

```
     APPE\EPP
```

Each time the key is pressed, the system deletes the rightmost chatacter. Display units actually erase each deleted character from the screen and move the printing position to the left.

In this example, the user presses CTRL/R (by typing R while the CTRL key is held down) to display the corrected text on a clean line (line 10) as follows:

```
APPE\EPP^R

A
```

The user then completes the line correctly and terminates it as usual with carriage return.

```
ACCEPT 2,K,L
```

If instead of CTRL/R the user had typed the amended letters CCEPT on the same line, the system would first have closed the string of deleted characters by a second backslash, thus:

```
APPE\EPP\CCEPT 2,K,L
```

On line 11, the DELETE key is used once more to delete the third 2.

```
2     FORMAT(22\2\I5)
```

2.4.2 Cancelling a Line

By mistake the user proceeds to type "PRINT" on the next line, but then presses CTRL/U to cancel the line and start again on line 13. CTRL/U (U pressed while the CTRL key is held down) deletes a line that has not been terminated by carriage return and advances the typing position to the beginning of the next line. The user can then enter the text that was originally intended.

```
PRINT^U

TYPE 3,K+L
```

CTRL/U is a useful way to correct a line whenever it is inconvenient to use the DELETE key.

2.4.3 Closing the New File

The last statement of the source program is "END" (line 16). After entering the last statement, the user types CTRL/Z (types Z while holding down the CTRL key) to indicate to the system that the file ADD.FTN is complete. The system displays ^Z and then prompts "PDS>" on the next line.

2.5 THE TYPE COMMAND

In response to the prompt (line 18) the user issues the TYPE command to display at the terminal the file ADD.FTN as it appears after corrections. The system responds by printing the contents of the file on lines 19 through 26.

2.6 THE FORTRAN COMMAND

After checking that the source program is correct, the user decides to run it. But the program must first be translated into instructions that the computer can understand. The translated source program is an "object module" of machine instructions.

In IAS, the FORTRAN command is used to translate a FORTRAN source program. So on line 27, the user types the following:

```
FORTRAN    ADD
```

In this case the user specifies the file as ADD rather than ADD.FTN. The FORTRAN command assumes the filetype to be FTN if it is not supplied.

After translating the program, the system prints the following text on line 28.

```
17:17:41  SIZE: 10K  CPU: 0.10
```

The figures "17:17:41" refer to the time at which the system finished translating the program. Line 28 also shows the amount of memory and CPU time used. "0.10" indicates that the translation required one tenth of a second compute time.

The system automatically places the translated FORTRAN program, now an object module, in a file named ADD.OBJ. (The filetype OBJ implies that the file contains an object module.)

2.7 THE LINK COMMAND

FORTRAN programs use a standard set of subprograms to perform certain functions. For example the FORTRAN statements TYPE and ACCEPT require the subprograms for input/output functions. The system maintains these subprograms in object module form so that they do not have to be translated each time someone uses them.

The purpose of the LINK command (line 29) in this sample session is to couple the object module contained in ADD.OBJ with the FORTRAN subprograms that it needs.

```
LINK    ADD
```

The omitted filetype is assumed to be .OBJ. If there is no file called ADD.OBJ, the system returns an error message. This might occur if a user tries to link an untranslated FORTRAN program, for instance.

Line 30 displays statistics about the completed execution of the LINK command.

The linked, executable program (the translated program linked with the required subprograms) is then placed in a file called ADD.TSK. The filetype TSK stands for "task" which is IAS terminology for an executable program.

2.8 THE RUN COMMAND

The FORTRAN and LINK commands have prepared the source program for execution. The user then issues the RUN command on line 31 to activate it.

```
RUN   ADD
```

Again, the filetype may be omitted. In this case the system assumes it to be .TSK. Line 32 shows the time the program began to run.

The FORTRAN program ADD is interactive; it requests the user to enter two numbers, then adds them together and displays the result (lines 33 to 35)

```
ENTER TWO NUMBERS

12,    78

THE SUM IS 9Ø
```

Writers of interactive programs must remember to prompt the user. If no prompts appear, the user cannot know what data to enter or at what point to enter it. This program uses the statements on lines 19 and 2Ø to display the prompt

```
ENTER TWO NUMBERS
```

The user supplies the numbers 12 and 78 on the next line and presses carriage return to terminate the input. The program then obeys the program statements on lines 23 and 24 by adding the numbers and declaring the sum to be 9Ø. The STOP statement (line 25) then causes the program to stop and the system to print the following line:

```
JOB16Ø -- STOP
```

The job number is the number assigned to the interactive session when the user logged in (see line 5).

The information displayed on the next line is similar to that on line 28 and 3Ø described in previous sections.

2.9 THE DIRECTORY COMMAND

In the session so far, the user has specifically created one file and caused the system to create two more, namely:

- ADD.FTN
- ADD.OBJ
- ADD.TSK

The system never automatically deletes a file, so all three must still exist. Only the system manager, the owner of the file or users authorized by the file owner can delete a file.

The DIRECTORY command (line 38) causes the system to display a list of the user's existing files. File information is stored in "directories". Line 39 identifies the user's directory as [2ØØ,22].

DIRECTORY DB0:[200,22]

The 200 identifies the user's group and the 22 identifies the user's number within the group. The text "DB0:" indicates that the directory resides on a volume mounted on a disk drive named DB0:

Line 40 states the date and time that the listing was requested.

The next three lines list the directory information:

```
     ADD.OBJ;1      2.       15-MAY-77   17:17
     ADD.FTN;1      1.       15-MAY-77   17:17
     ADD.TSK;1      32.    C 15-MAY-77   17:18
```

Notice that ";1" appears at the end of each file name. The number 1 is the file's version number and indicates that each file listed is the first version of the file. If the user were to issue the command FORTRAN ADD again, the FORTRAN translator would produce a second object file called ADD.OBJ;2. Users can either delete old versions or retain them as security against the loss of later versions.

The value in the second column indicates the number of 512 byte blocks occupied by each file on the disk. The date and time show when each file was created. The "C" that appears on the third line between the number of blocks and the date declares that the blocks within ADD.TSK;1 are "contiguous"; that is, they are physically located one next to the other.

2.10 THE RENAME COMMAND

The RENAME command allows the user to change the name of a file without changing its contents or location. The user now issues the command to rename all three files named ADD at the same time (line 45)

```
     RENAME    ADD.*;*    ADDTWO.*;*
```

The asterisks (*) that appear in the above line are the mechanism that allow the user to specify all three files at once. An asterisk or "wild-card", is a shorthand notation for "all". ADD.*;* means all the files that have ADD as a filename, disregarding the filetype and version number. In this case, ADD.*;* refers to the files ADD.FTN;1, ADD.OBJ;1 and ADD.TSK;1. The user could also refer to these three files in the following manner:

```
     ADD.*;1
```

since all the files have the same version number but different filetypes. The command issued on line 48 changes the files' name from ADD to ADDTWO. The wild-cards in the text "ADDTWO.*;*" mean that the renamed files retain their original types and versions. The files are now called

- ADDTWO.FTN;1
- ADDTWO.OBJ;1
- ADDTWO.TSK;1

2.11 THE DIRECTORY/BRIEF COMMAND

When the user reissues the DIRECTORY command (lines 46 and 47) the system lists the files with their new filenames. (Note that CTRL/U was pressed to cancel line 46 because of a typing error. See the description of CTRL/U in Section 2.4.2).

This instance of the DIRECTORY command includes the text "/BRIEF", a "qualifier" which modifies the action of the command. /BRIEF causes the system to list only the names of the files and to omit information about blocks and time of creation.

Most commands have one or more qualifiers. A slash (/) always precedes the qualifier's name. When a user specifies more than one, the slashes separate one from the next.

2.12 THE LOGOUT COMMAND

To end the interactive session, the user issues the LOGOUT command (line 52). The system then displays user and accounting information on the next two lines and the text "BYE" on the third line.

The terminal is now inactive and CTRL/C must be pressed to invoke PDS once more.

CHAPTER 3

KEYBOARD OPERATION

The purpose of this chapter is to acquaint the user with the keyboard layouts of interactive terminals and to describe the keyboard functions and how to use them under IAS. Instructions on how to log into the system and to use PDS are contained in Chapter 4.

3.1 THE KEYBOARD

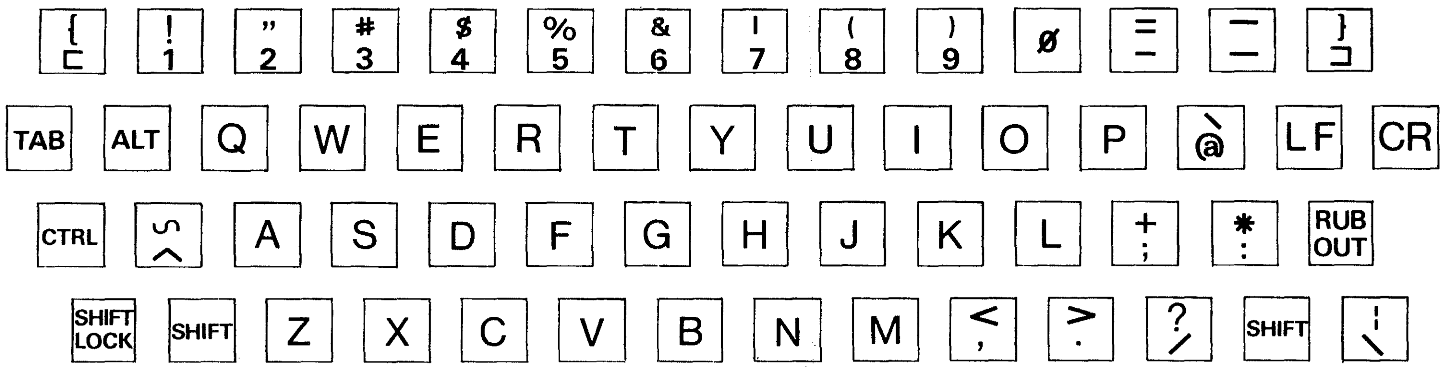
The interactive user types data directly into the system from a terminal (for example, a DECwriter or a display unit) instead of supplying input data on punched cards or paper tape. The keyboard layout of an interactive terminal is very similar to the layout of an ordinary typewriter. The number and letter keys are in the traditional typewriter format, but punctuation marks, special characters and function keys may differ in position from one type of terminal to another (see Figures 3-1 and 3-2).

3.1.1 Keyboard Functions

The user types the input text one line at a time, terminating each line with carriage return (CR or RETURN) or altmode (ALT or ESC). The system either prints the terminal input on the terminal printer or displays it on the screen of a display unit (except when the user types a password, see Chapter 2, Section 2.3.1.2).

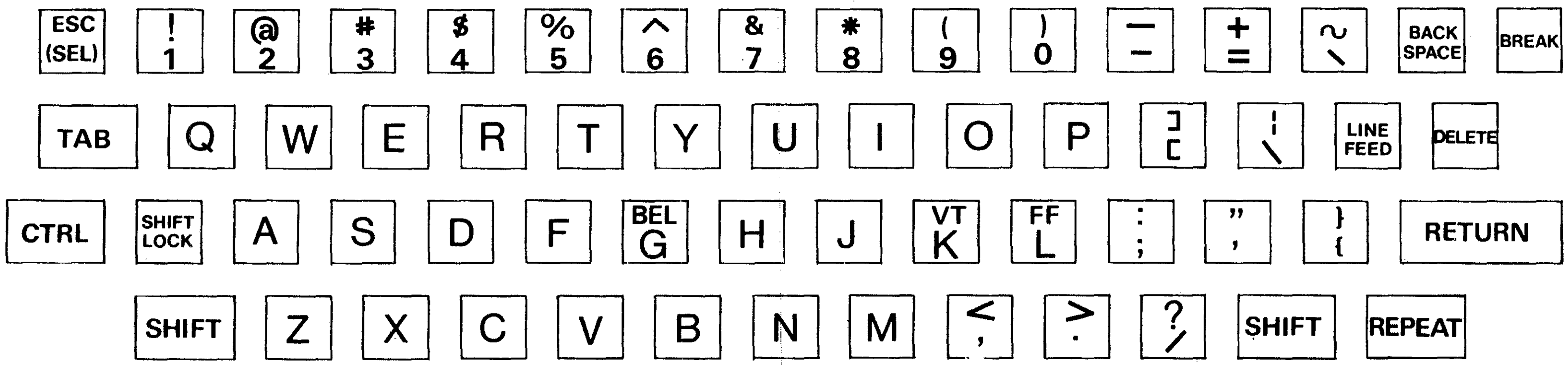
Function keys can be used to format a line (Space Bar, TAB), to edit a line (RUBOUT/DELETE), or to access the uppermost of two characters that appear on a key (SHIFT, SHIFT LOCK). The CTRL key, when pressed simultaneously with a letter key, provides further keyboard functions; these functions are described in detail in Section 3.1.2. Typing a carriage return (CR or RETURN) causes the system to store the current line or to carry out some specified action.

Table 3-1 describes the function keys and the effects of their use under IAS.



{ [! 1 " 2 # 3 $ 4 % 5 & 6 ' 7 (8) 9 Ø = - — — }]
TAB ALT Q W E R T Y U I O P ` @ LF CR
CTRL ~ ^ A S D F G H J K L + ; * : RUB OUT
SHIFT LOCK SHIFT Z X C V B N M < , > . ? / SHIFT | \

Figure 3-1
LA30/VT05 Layout



ESC (SEL) ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 — - + = ~ ` BACK SPACE BREAK
TAB Q W E R T Y U I O P] [| \ LINE FEED DELETE
CTRL SHIFT LOCK A S D F BEL G H J VT K FF L : ; " ' } { RETURN
SHIFT Z X C V B N M < , > . ? / SHIFT REPEAT

Figure 3-2
LA36/VT50 Layout

Table 3-1
Keyboard Functions

Key	Description
CR or RETURN	Carriage return. Transmits the current line to the computer and performs a carriage return line feed. When keyed after a PDS command string, causes PDS to issue the next prompt for mandatory input. PDS omits intervening prompts, if any, for optional input.
CTRL	Is part of several 2-key combinations that produce a variety of functions. See Section 3.1.2.
DELETE RUBOUT	Deletes the last typed character. May be used repeatedly. On a display unit, the current printing position moves to the left and the deleted character is erased. On other terminals the string of deleted characters is echoed between an initial backslash (\) and a final backslash (\). See Section 3.2.
ESC or ALT	When keyed after a PDS command string, it causes PDS to prompt for the next input, whether optional or mandatory. This character can be echoed as $, depending on the installation.
LINE FEED or LF	Has no control effect under IAS.
SHIFT	Prints or displays the uppermost of two characters appearing on a key typed while SHIFT is held down. SHIFT has no effect when used with keys that have only one character.
SHIFT LOCK	Alternately locks and unlocks SHIFT mode.
SPACE BAR	Advances the current typing position one space at a time.
TAB	Causes the current typing position to move to the next tab stop on the line. A line conventionally contains tab stops every 8 spaces.

3.1.2 Control Key Functions

Typing a character key while pressing the control key (CTRL) invokes one of the functions listed in the following table. The combination of CTRL and another character key is called a control character. In this manual a control character is written "CTRL/X" where X is the variable character key.

The effect of a control character sometimes depends on the activity that the terminal is currently supporting.

Table 3-2 lists the control characters supported under IAS and their associated functions.

Table 3-2
Control Key Functions

Control Character	Function
CTRL/C	Before a user has logged in, invokes PDS. and returns control to PDS. CTRL/C will terminate the DIRECTORY and DELETE commands. Cancels a command if issued between the PDS> prompt and carriage return.
CTRL/B	On a terminal set with low-speed paper tape reader support, CTRL/B signals to the computer to start reading the tape, the reader being already switched on.
CTRL/I	Causes the current typing position to move to the next tab stop on the line. Performs the same action as the TAB key.
CTRL/K	Advances the current line to the next vertical tab stop. Equivalent to a Line Feed.
CTRL/L	Advances continuous stationery to the next top of form. Equivalent to a Form Feed.
CTRL/O	Interrupts and suppresses output to the terminal. Successive pressings of CTRL/O cause output to be suppressed and to resume. For example, if a directory listing on the terminal is requested and the first few lines present the desired information, CTRL/O can suppress the rest of the directory.

(continued on next page)

Table 3-2 (Cont.)
Control Key Functions

Control Character	Function
CTRL/Q CTRL/S	These two keys correspond to 'transmission on' (XON) and 'transmission off' (XOFF) respectively. Pressing CTRL/S (XOFF) stops output to the terminal until CTRL/Q (XON) is pressed. Unlike CTRL/O, the XOFF/XON function stops and starts output without any loss of characters.
CTRL/R	Retypes the current line with any deleted characters removed. See Section 3.2.2.
CTRL/T	On a terminal set with low-speed paper tape reader support, CTRL/T stops a read. CTRL/T can be present on the tape, or the reader can be switched off and then CTRL/T typed at the terminal.
CTRL/U	Deletes the current input line. The prompt, if any, is then repeated. See Section 3.2.3.
CTRL/V	Typing CTRL/V flushes all characters typed ahead of a read. If a read is in progress CTRL/V has no effect. For type-ahead modes see the IAS/RSX-11D Device Handlers Reference Manual, Chapter 2.
CTRL/Z	Terminates a file input from a terminal, that is, signals "end of file".

3.2 CORRECTING INPUT ERRORS

Before terminating a line, the user can correct typing errors or change the line completely by using RUBOUT or DELETE or CTRL/U. However, once the line has been terminated and thus transmitted to the computer, it can be corrected only by means of an editing program.

3.2.1 Cancelling a PDS Command

Typing CTRL/C cancels a PDS command that has not yet been terminated.

3.2.2 Deleting Individual Characters

The DELETE or RUBOUT key deletes the most recent character on the current line for each pressing of the key. DELETE has no effect when the current line is empty.

On a hard-copy terminal, each deleted character is echoed. The string of deleted characters is enclosed between an initial and a final backslash (\). The final backslash is added when a new text character is typed in place of DELETE. It is omitted in the case when CTRL/R is used to make a 'fair copy' of the line as typed so far (Section 3.2.3).

On a Visual Display Unit (VDU) each deleted character is removed from the screen, and the cursor returns to where it was before the character was typed.

For example, to change ACCDE to ABCDE, the user presses DELETE or RUBOUT four times to override the CCDE. On a hard-copy terminal the string now appears as

```
ACCDE\EDCC
```

The user then enters the correct sequence BCDE. On the hard-copy terminal, the string now appears as

```
ACCDE\EDCC\BCDE
```

On a display unit the screen will show the string

```
ABCDE
```

In both cases ABCDE is the string accepted and sent to the computer when the line is terminated.

3.2.3 Deleting a Line

CTRL/U deletes all characters on the line, prints ^U and performs a carriage return. The user can then enter the text correctly.

For example, if a user types ACCDEFGHI, but meant to type B for the first C, pressing the RUBOUT key eight times would be tedious and the result confusing on a hard copy terminal. It would be easier to press CTRL/U and start again. The latter solution would appear as follows:

```
ACCDEFGHI ^U
ABCDEFGHI
```

After using the RUBOUT or DELETE key to correct a line and before terminating the line, the user can ensure that the final result is in fact correct. To display the line as it will be sent to the computer, simply press CTRL/R. With CTRL/R and CTRL/U the prompt, if any, is repeated.

Further corrections can be made at this point if necessary.

3.3 USE OF UPPER AND LOWER CASE

On terminals that are equipped with upper and lower case letters, PDS commands may be entered in either case. In general, lower case characters can be converted, and echoed, to upper case, depending on the characteristics currently defined for the terminal. The conversion can also be performed for individual tasks as required. See the IAS/RSX-11D Device Handlers Reference Manual, Chapter 2.

CHAPTER 4

ISSUING PDS COMMANDS

4.1 COMMAND NAMES AND PARAMETERS

The user communicates with the system via PDS, by issuing commands at an interactive terminal or by submitting a file of commands to a batch queue. A command consists of a command name which describes the action the system is to take (COPY or LOGIN, for example), usually accompanied by one or more parameters. Parameters either describe the items on which the command is to act or further define the function of the command.

Commands can only be entered at an interactive terminal when the system is prompting "PDS>". Some PDS commands (EDIT and BASIC, for example) invoke a program that accepts its own set of commands, valid only while that program is running. In turn, PDS commands are not valid while that program is running; the user must first return control to PDS. The specifications of EDIT and BASIC in Part 2 describe how to terminate the invoked program's execution.

4.1.1 Command Strings

Batch command strings contain the command name and parameters in a single or continued line. Interactive users can either supply the command name followed by the parameters on one line or enter the parameters in response to prompts (see Section 4.1.3 below). In both batch and interactive mode, when two or more parameters are on one line, they must be separated by a comma, spaces and/or tabs.

If a command runs to more than one line, a hyphen (-) as the last character on the line or card causes the command to be continued onto the next line.

An exclamation mark (!) after the last character of any command line indicates the start of a comment. The comment text appears after the exclamation mark.

4.1.2 Parameters

The parameters to the COPY command (see Chapter 6, Section 6.4.2.2), which are an input file specification and an output file specification, can be input in any one of the following ways.

In interactive mode:

```
     1.   PDS> COPY RISE.MAC WORK.MAC

     2.   PDS> COPY RISE.MAC , WORK.MAC

     3.   PDS> COPY

          FROM?  RISE.MAC WORK.MAC

     4.   PDS> COPY RISE.MAC

          TO?  WORK.MAC

     5.   PDS> COPY

          FROM?  RISE.MAC

          TO?  WORK.MAC
```

In batch mode:

```
     1.   $COPY RISE.MAC   WORK.MAC

     2.   $COPY RISE.MAC,WORK.MAC

     3.   $COPY RISE.MAC,   WORK.MAC
```

4.1.3 Parameter Prompts

The LOGIN command demonstrates how PDS prompts for command parameters at an interactive terminal (See Chapter 2, Section 2.2). The prompting facility greatly minimizes input errors by interactive users who are unsure of the command parameters.

The more experienced user may be very familiar with the commands and not need the prompts. PDS therefore suppresses prompts for parameters that are included on the previous line. For example, the LOGIN command may be input as follows:

```
     PDS> LOGIN WILSON

     PASSWORD?
```

Because the User Name (WILSON) was typed on the same line as LOGIN, separated from the command by a space, PDS suppresses the prompt USER-ID? and displays the next one, i.e. PASSWORD?.

NOTE

The user should not type the password on the same line as the LOGIN command so that it is not echoed on the terminal.

4.1.4 Optional Parameters

Interactive PDS commands prompt for both mandatory and optional parameters. To display the prompt for an optional parameter, however, the user must use ALTmode (ESCape) rather than carriage return after the last mandatory parameter. For example:

```
PDS> MOUNT <CR>

DEVICE? DK· <CR>

VOLUME-ID? CHARLY <ALT>

LOGICAL NAME? AB
```

where LOGICAL NAME? is an optional prompt.

To suppress the prompt LOGICAL NAME?, the user must press carriage return after CHARLY. For example:

```
PDS> MOUNT DK2:  CHARLY <CR>
```

NOTE

> Carriage return and ALTmode have the same effect on a command line when not used immediately before an optional prompt.

If an optional prompt has been invoked by mistake, simply press carriage return immediately after the prompt. For example:

```
PDS> MOUNT DK2:  CHARLY <ALT>

LOGICAL NAME?  <CR>
```

Batch users may either omit the optional parameter from the command string if it is the last parameter, or replace the optional parameter with two commas if there are further parameters to be specified.

4.1.5 Parameter Lists

Some parameters may be replaced by a list of parameters enclosed in parentheses and separated by spaces, tabs and/or a comma. Parentheses are not required, however, when the list replaces a parameter that is the last or only parameter in the command. Examples:

1. `PDS> APPEND (FILEA.FTN,FILEB.FTN) FILEC.FTN`

2. `$DELETE AB.CBL;1, AB.OBJ;1`

4.2 ABBREVIATED INPUT

A user only needs to enter enough of a command to distinguish it from all other PDS commands. All command names can be uniquely abbreviated to four letters.

For example, the LOGIN command may be shortened to:

```
    LOGI
```

and still be accepted by the system; but LOG is not acceptable because it does not distinguish LOGIN from LOGOUT.

4.3 COMMAND AND FILE QUALIFIERS

The command string

```
    PDS> PRINT/DELETE
```

is an example of the PRINT command (see Chapter 6, Section 6.4.3.1). The command requests the system to output on the line printer the file specified on the next line, and to delete the file after it has been printed.

Command qualifiers modify the function of the command. The main purpose of the PRINT command is to output one or more specified files on a line printer. To delete the file or files is an option that the user indicates by specifying the command qualifier /DELETE.

For example, the qualifiers to the FORTRAN command (see Chapter 11 Section 11.2), which invokes the FORTRAN compiler, determine the form of the output generated by the compiler.

Each qualifier may be abbreviated by supplying enough characters to distinguish it from any other possible qualifiers.

File specifications may also have qualifiers; these qualifiers describe properties the file has or is to have. For example, the /PROTECTION qualifier may modify the file specification supplied with the CREATE command (see Chapter 6, Section 6.4.1.2). The qualifier determines the protection code applied to the newly-created file. Example:

```
    $CREATE NEWFILE.DAT/PROTECTION:(SY:RWED, OW:RWED,GR:R, WO:R)
```

4.3.1 Underbar Convention

To increase legibilty, some qualifiers have an underbar character where two or more English words have been joined together, for example:

```
    PDS> MOUNT/FILE_PROTECTION:(code)
```

When such qualifiers are abbreviated, the underbar is treated in the same way as the alphabetic characters. Thus

```
    PDS> MOUNT/FILE_PROT:(code)
```

or

```
     PDS> MOUNT/FI:(code)
```

are acceptable, since they determine this qualifier uniquely among the MOUNT qualifiers.

The underbar convention does not apply to the prefix NO.

4.4 UNACCEPTABLE COMMANDS OR SYNTAX

There are many reasons why PDS may not be able to execute a command.

4.4.1 Effect of Tasks Run from a Terminal

In IAS terms, a running program is called a "task". The IAS Executive Reference Manual, Volume One describes tasks in detail.

When a task is running from an interactive terminal, the user may not issue any PDS commands until the task has terminated or been suspended. To suspend the task, the user must press CTRL/C. The user might then issue the SHOW STATUS command to check on the progress of the task. Depending on the information displayed, the user would either issue the ABORT command to abandon the task or the CONTINUE command to resume execution.

Most PDS commands cannot be issued while a task is suspended. If the user tries to issue an unacceptable command, IAS displays the message:

```
     COMMAND NOT ALLOWED    SUSPENDED TASK
```

The user must either issue ABORT to abandon the task or CONTINUE to resume it.

4.4.2 Subsystems

PDS commands are not valid when the user is operating within a subsystem such as BASIC or the Line Text Editor. The user must first return control to PDS and then issue a PDS command.

4.4.3 Error Messages

When a command fails, PDS displays an error or diagnostic message that indicates where the problem lies.

The following interactive session includes examples of command failures and the resultant system responses. Asterisks have been added to responses that indicate failure of one sort or another.

```
    PDS> LOG

   *PLEASE LOGIN (OR TYPE HELP)

    PDS> LOGI

    USER NAME?  SMITJ
```

```
  PASSWORD?  (The terminal does not display the password)

 *USER NAME NOT AUTHORIZED

  PDS> LOGI SMITH

  PASSWORD? (The terminal does not display the password.)

 *PASSWORD?

 *PASSWORD?

  USER SMITH   UIC [1ØØ,1ØØ] TTØ7 JOB-ID 4Ø TIME 16:29:1Ø  15-MAY-77

  PDS> COPY

  FROM? A$B

 *A - ILLEGAL FILE-SPECIFICATION

  PDS> DIRECTORY <ALT>

  FILE? A:B

 *A  -  ILLEGAL DEVICE

 *ILLEGAL FILE-SPECIFICATION
```

The reasons for failure are as follows:

1. PLEASE LOGIN (OR TYPE HELP) - The user did not type enough of the command to make it unique. The system could not tell whether LOG was a shortened form of LOGIN or LOGOUT.

2. USER NAME NOT AUTHORIZED - The User Name (SMITJ) supplied did not grant the user access to PDS because the user had mistyped the last character.

3. PASSWORD? - By repeating the password prompt, the system indicated that the user SMITH had not typed the correct password (see Section 3.3.1.2).

4. A - ILLEGAL FILE-SPECIFICATION

   "$" is not a valid character within a file specification.

5. A - ILLEGAL DEVICE

   ILLEGAL FILE-SPECIFICATION

   "A" is not a valid IAS device name.

Common errors include:

- mistyping characters within a command
- not leaving a space where it is needed to distinguish between command components

- providing parameters in an incorrect order
- specifying incorrect devices

4.5 PDS COMMAND PRIVILEGE

PDS Command Privilege governs the right of an individual user to issue a specific command or set of commands via PDS.

These rights are given or withheld by the system manager when the user is authorized.

4.5.1 PDS Command Masks

Each user is allocated two PDS Command Masks on authorization. One concerns interactive terminal use and the other batch use. Each mask consists of 16 bits. A bit is set to 1 to make the corresponding command(s) available. The bits are referred to by symbolic names. These are used in the following two tables.

Table 4-1

PDS Command Privilege Classes

Bit	Symbol	Command or Class of Commands
0	PR.FIL	File manipulation facilities
1	PR.RUN	Task manipulation
2	PR.BAS	BASIC
3	PR.COB	COBOL
4	PR.COR	CORAL
5	PR.FOR	FORTRAN
6	PR.LIN	LINK
7	PR.MAC	MACRO
8	PR.400	Not allocated
9	PR.SUB	SUBMIT (to Batch)
10	PR.RES	Reserved
11	PR.DEV	Device management
12	PR.DUM	DUMP
13	PR.LIB	LIBRARIAN
14	PR.SYS	System library tasks ($$$xxx)
15	PR.RTC	Real time commands

Certain commands are available to all logged-on users. These are marked ANY in Table 4-2, and are independent of the command masks.

The privilege required for each command is shown in Table 4-2. A few commands require two privileges. A few require that the user has an associated UIC of [1,1] and are so marked. Of these, the USERS command is described in the IAS System Management Guide.

Table 4-2
PDS Command Privileges Required

Command	PDS Command Privileges Required
ABORT (timesharing)	ANY
ABORT/REALTIME	PR.RTC
ALLOCATE	PR.DEV
APPEND	PR.FIL
ASSIGN (timesharing)	PR.RUN
ASSIGN/REALTIME	PR.RUN, PR.RTC
BASIC	PR.BAS
CANCEL	PR.RTC
COBOL	PR.COB
COMPARE	PR.FIL
CONTINUE (timesharing)	ANY
CONTINUE/MESSAGE	PR.RTC
CONTINUE/REALTIME	PR.RTC
COPY	PR.FIL
CORAL	PR.COR
CREATE	PR.FIL
DEALLOCATE	PR.DEV
DEASSIGN	PR.DEV
DELETE	PR.FIL
DIRECTORY	PR.FIL
DISABLE	PR.RTC
DISMOUNT	PR.DEV
DUMP	PR.DUM
EDIT	PR.FIL
ENABLE	PR.RTC
EOJ	N/A
FIX	PR.RTC
FORTRAN	PR.FOR
GOTO	ANY
HELP	N/A
INITIALIZE	PR.DEV
INSTALL	PR.RTC
JOB	N/A
LIBRARIAN	PR.LIB
LINK	PR.LIN
LOGIN	N/A
LOGOUT	N/A
MACRO	PR.MAC
MERGE	PR.FIL
MESSAGE	ANY
MOUNT	PR.DEV
ON	ANY

(continued on next page)

Table 4-2 (Cont.)
PDS Command Privileges Required

Command	PDS Command Privileges Required
PRINT	PR.FIL
QUEUE	PR.FIL
REMOVE	PR.RTC
RENAME	PR.FIL
RUN (timesharing)	PR.RUN
RUN (realtime)	PR.RUN, PR.RTC
SET DEFAULT	ANY
SET [NO]QUIET	ANY
SET PASSWORD	ANY
SET PRINTING	ANY
SET PRIORITY	PR.RTC
SET PROTECTION	PR.FIL
SET TERMINAL attribute	ANY
SET TERMINAL:TTn attribute	[1,1]
SET TERMINAL:(TTm,...,TTn) attribute	[1,1]
SHOW CLI	ANY
SHOW DAYTIME	ANY
SHOW DEFAULT	ANY
SHOW DEVICES (other than /PUD)	ANY
SHOW DEVICES /PUD	[1,1]
SHOW GLOBAL_AREAS	ANY
SHOW LUNS	PR.RTC
SHOW MEMORY	ANY
SHOW PARTITIONS	ANY
SHOW STATUS	ANY
SHOW TASKS	ANY
SORT	PR.FIL
STOP	ANY
SUBMIT	PR.SUB
TYPE	PR.FIL
UNFIX	PR.RTC
UNLOCK	PR.FIL
USERS	[1,1]

4.6 PDS TIMESHARING TASK PRIVILEGE

IAS can also restrict the execution of timesharing tasks.

A task can be built to map part of its virtual address space on to SCOM, the IAS System Communications Area. Such a task is said to be 'executive privileged'. (See the IAS Task Builder Reference Manual, under LINK/PRIVILEGED, and the IAS Executive Reference Manual Volume II).

Also, a program can be written using the directives noted as privileged in the Executive Reference Manual, Volume II. The resulting task is then said to be 'directive privileged'.

PDS Users who wish to run such tasks in timesharing need certain bits to be set in their PDS Timesharing Task Privilege Mask. This mask is set up by the system manager when the user is authorized.

If a PDS user tries to initiate a task that is executive privileged, and the user is not authorized to run such tasks, the task is rejected before execution begins.

If a PDS user runs a task in timesharing that issues privileged directives, and the user is not authorized with directive privilege, each privileged directive is rejected as met but the task continues to execute.

Note that all tasks executing in real-time mode are given directive privilege and can be executive privileged, independently of the initiating user's timesharing task privilege mask.

4.7 PDS DIALUP SUPPORT

If a dialup line is lost during an interactive session while the user is logged in, the job is not lost but remains attached to the same line. If a task is running it will be suspended. If any user dials up and is connected to the same line, the following message is printed, followed by a PDS prompt:

USER username ALREADY LOGGED IN [WITH SUSPENDED TASK]

where

username	is the name of the user currently logged in

At this point, only two commands are valid:

LOGOUT	to logout the user and free the terminal
CONTINUE	to enter normal interactive mode. If there is a suspended task, it will be continued and it will be necessary to type CTRL/C to suspend it again. Before continuing, PDS prompts 'PASSWORD?' and checks the user's password.

If no user connects to the suspended line within the timeout limit set for PDS by the System Manager, the user will be logged out and the line disconnected. This also happens if the suspended task exits for any reason while the line is suspended (for example, if it completes a Mark Time directive and exits).

If the task which is running attempts to perform input or output to the terminal during the short period (about one second) between the loss of the dialup line and the suspending of the task, it will receive an error. Some commands (for example, DIRECTORY) will terminate.

CHAPTER 5

BATCH PROCESSING

5.1 INTRODUCTION

Almost all IAS commands are applicable to both interactive and batch processing. Batch users, however, begin and end a job with the $JOB and $EOJ (End of Job) commands, rather than with LOGIN and LOGOUT (see Chapter 2). Batch commands must always begin with a dollar sign ($) in the first position of a line.

Batch users may submit a job either:

1. From an interactive terminal, or
2. Via a card reader

The first method requires the PDS command SUBMIT, which submits a file of batch commands to the batch processor. The processor queues the submitted job until all the jobs preceding it in the queue have terminated. See Section 5.3 for a full description of the SUBMIT command.

When submitting a job via a card reader, the user includes the batch commands in the input stream.

For example:

```
$JOB GRAHAM CATJOB 3
$COBOL JOB.CBL
$EOJ
```

This example invokes the COBOL compiler to compile the source program held in the file JOB.CBL.

5.2 BEGINNING AND ENDING A BATCH JOB

The $JOB and $EOJ commands delimit a single batch job.

5.2.1 The $JOB command

The $JOB command marks the beginning of a batch job. Parameters to the command consist of, in the following order, the User Name, a job name and a time limit in minutes for the job's elapsed time.

For example:

```
$JOB CATHY TEST 3
```

CATHY is the User Name and TEST is the job name. The number 3 instructs the system to terminate the job after it has used 3 minutes of elapsed time.

The User Name is a 1- to 12-character alphanumeric string that is unique to the individual user; it is identical to the User Name parameter to the LOGIN command (see Chapter 2, Section 2.3.1.1.) and must be a current USER-ID in the system.

The job name is a 1- to 12-character alphanumeric string that identifies the job.

For jobs that are to be run for users who have restricted batch access to their user-name, the $JOB command may also take an optional PASSWORD qualifier:

For example:

```
     $JOB/PASSWORD:SECRET SYSTEM ACCOUNTS 30
```

5.2.2 The $EOJ Command

The $EOJ command terminates a batch job. It has no parameters.

5.3 THE SUBMIT COMMAND

The SUBMIT command submits a file of batch commands to a batch queue from an interactive terminal. When batch is activated to process entries from the batch queue, it begins with existing queue entries and then processes any jobs submitted while it is still active.

For example:

```
     PDS> SUBMIT BATCHJOB.CMD
```

Submit the file BATCHJOB.CMD to the PDS batch processor.

See Chapter 6, Section 6.4.1, for instructions on creating a file to contain the batch commands.

5.4 BATCH EDITING

IAS provides a batch-oriented editor to create and maintain source language files and data files on disk. This editor, called the Source Language Input Program (SLIPER), is described in Chapter 7.

CHAPTER 6

FILE HANDLING

6.1 INTRODUCTION

All the information that is stored in a computer system is held in logical units called files. A file is defined as an ordered collection of information. In order to store information, a source program, for instance, a user must create a file and input the source program to it.

Any subsequent attempts to access or manipulate the source program must be made in terms of the file that contains it, that is, by supplying a file specification. A file specification gives the system all the details it needs to identify the file: the device on which it is stored, the directory of the file, the file name, the filetype and the version.

This chapter describes IAS file handling commands and how to use them.

6.1.1 IAS File System

The standard IAS file system for disks, DECtapes and magnetic tapes is the Files-11 system. Files-11 magnetic tapes conform to American National Standard Magnetic Tape Labels and File Structure for Information Interchange, X3.27-1969. A detailed description of the Files-11 file system is contained in the IAS Executive Reference Manual - Volume II and the IAS/RSX I/O Operations Reference Manual. Most PDS commands can only operate on Files-11 files.

6.1.2 Volumes

The magnetic media on which files are stored are called volumes, for example, disks, magnetic tapes. In order to access a file held on a volume, that volume must be mounted, that is, physically loaded on a disk or tape drive and related to the user's task or terminal by the MOUNT command (see Section 6.3.1). Volumes that do not hold files in Files-11 format must be mounted using the qualifier /FOREIGN.

6.1.3 Volume and File Protection

IAS protects the individual user's privacy and the system's security by providing a facility to restrict access to a volume. Magnetic tapes written in Files-11 format have a volume level protection code; that is, the protection assigned to the volume applies equally to every file within it. Disks and DECtapes, however, have both an overall protection code for access to the volume and individual protection codes for each file within it.

For the purposes of assigning protection codes, IAS defines four types of access, read (R), write (W), extend (E) and delete (D), and four categories of user, system, owner, group, world. The protection code designates the kind of access each user category is allowed. The user categories are defined as follows:

User	Description
SYSTEM:	All tasks that run under a system User Identification Code (UIC).
OWNER:	All tasks that run under the UIC of the owner of the file or volume.
GROUP:	All tasks that run under a UIC that has the same group number as the UIC of the owner of the file or volume.
WORLD:	All tasks.

The system uses the User Identification Code to determine file ownership. The system identifies a user's UIC from his User Name. The code is not necessarily unique to each user.

Volume protection is applied when the volume is initialized by the IAS system manager and can be re-specified via the MOUNT command (see the specification of MOUNT in Part 2).

A file's protection code is applied when the file is created and the code may subsequently be modified by the SET PROTECTION command. If the user does not explicitly specify a protection code for a newly-created file, the system automatically applies the volume's default code.

Example:

```
PDS> SET PROTECTION

FILE? MYFILE.DAT

PROTECTION? (SYS:RWED, WO:, G:RW)
```

The example above changes the protection code of the file MYFILE.DAT so that the system (SYS:) has all four types of access, the world (WO:) is denied all types of access, the group (G:) has read and write access, and the allowed access of the owner does not change. This example illustrates the following rules:

1. The protection code must always be enclosed in parentheses.

2. The four user categories are represented by codes followed by colons. The codes may be abbreviated to one or more letters.

The codes are:

SYSTEM:

OWNER:

GROUP:

WORLD:

3. The four types of access are represented by single letters:

R	Read
W	Write
E	Extend
D	Delete

4. Each category that is mentioned is allocated the types of access specified after the code and denied any type of access not specified; for example, GR:RW gives group members read and write access only. If no types of access are specified after a category, all types of access are denied to it, for example, WO:

5. Any category not mentioned keeps the access privileges previously allocated to it.

6. The user categories and types of access may be specified in any order.

6.1.4 RMS-11 Files Management in IAS

Digital's Record Management System is supported by IAS. RMS-11 is a suite of routines for managing three types of file organization. The three differ in the way the records within a file are accessed. The record is the basic unit of information handled by RMS-11. Examples are the input at a terminal delimited by carriage returns, or the contents of a single punched card.

'Sequential' is the default organization in IAS. To find a particular record, for example when using the IAS editor, each record must be accessed in sequential order until the required one is located.

'Relative' is an organization by block number and allows individual records to be accessed directly and randomly.

'Indexed Sequential' or, more shortly, 'Indexed' allows records to be handled, copied or sorted depending upon the contents of the record in previously specified fields (KEYS).

These concepts are described in the Introduction to RMS-11 manual. They are implemented for IAS by the PDS commands:

APPEND	CREATE
COPY	MERGE

As described in Part 2. For the interface at program level see the IAS/RSX-11M RMS-11 MACRO Programmer's Reference Manual.

6.2 FILE SPECIFICATIONS

A file specification provides the system with all the details it needs

- to create a file
- to identify an existing file stored on a volume
- to read a file from or write a file to a device such as a line printer or a card reader

The basic format of a file specification is as follows:

```
dev:[ufd]name.typ;ver
```

where

dev:	is a device name of the form XXnn: where XX is a 2-letter mnemonic for the device (see Table 6-1) and nn is a 1- or 2-digit octal number from Ø to 77. The device mnemonics are listed in Table 6-1. The device field may be replaced by a logical name (see Section 6.3.1).
[ufd]	is the UFD (User File Directory) of the form [m,n] where m and n are octal numbers from 1 to 377.
name	is the name of the file, an alphanumeric character string from 1- to 9-characters long.
typ	is a 1- to 3-alphanumeric character filetype that usually identifies some aspect of the file contents. Table 6-2 lists standard filetypes for IAS files. For example, the filetype FTN indicates that the file contains a FORTRAN source program.
ver	is the version number, an octal number in the range 1 to 77777 used to differentiate among versions of the same file. For example, when a file is created, the system assigns the file a version number of 1. If that file is subsequently opened for editing, the editor retains the original file for backup by creating the new file with the same filename and type, but with a version number of 2.

Table 6-1 lists the 2-character mnemonics conventionally used in the device name field of file specifications.

Eight of these mnemonics, namely CO, LB, MO, SP, SY, TI, TO and WK, are logical device names, ("pseudo-devices"), which can be made to refer to particular physical devices according to the needs of the computer installation.

Table 6-1
IAS Device Types

Mnemonic	Device Type
AD	ADØ1 A/D converter
AF	AFC11 Analog input
CO	Console output
CR	Card reader
CT	Cassette
DB	RPØ4 disk
DF	RF11 disk
DK	RKØ5 disk
DM	RKØ6 disk
DP	RPØ2 or RPØ3 disk
DS	RSØ3 or RSØ4 disk
DT	DECtape
DX	Floppy disk
LB	Device holding system library files
LP	Line printer
LS	LPS A/D converter
MM	TU16 magnetic tape
MO	Message output
MT	TU1Ø magnetic tape
SP	Device holding spooled I/O files
SY	User's system disk
TI	User's data input stream
TO	User's data output stream
UD	UDC11 Universal Digital Control
WK	Fast-access device for work files.

TI and TO are logical device names for a user's input and output data streams. For example, when a user wishes to read from his terminal he specifies TI:

```
     PDS> COPY

     FROM?  TI:

     TO?  MYFILE.DAT
```

transfers the input text typed at the user's terminal to the file named MYFILE.DAT.

Table 6-2 lists all the standard IAS file types.

Table 6-2
Standard IAS File Types

File Types	Description
BAS	A BASIC language source file
BIS	A batch command file
CBL	A COBOL language source file
CMD	A file containing a list of commands (indirect file)
COR	A CORAL language source file
DAT	A data file
DIR	A directory file
FTN	A FORTRAN language source file
LST	A file in print-image format
MAC	A MACRO-11 assembly language source file
MAP	A file containing a memory allocation map
MLB	A macro library file
OBJ	An object program (output from MACRO-11 or FORTRAN)
ODL	An overlay description file
OLB	An object library file
SAV	A saved system memory image file
SML	A system macro library file
SPR	A spooled output file
SRT	A SORT specification file
STB	A symbol table file
TMP	A temporary file
TSK	A task image file produced by the Task Builder and suitable for execution.

6.2.1 Defaults

A user may omit the device name and/or the UFD field of any file specification. In this case, the system replaces the null fields with the user's default values.

The version number may also be omitted, in which case, the system assumes:

1. The highest version number for an input file specification or

2. The highest version increased by one for an output file specification or 1 if no previous version exists.

The device and UFD defaults are determined initially as follows:

1. The default device is determined for each user by the system manager.

2. The default UFD is equivalent to the UIC (see Section 6.1.3) associated with the user's User Name (submitted at log in).

The following table lists the default values, if any, of the various fields.

Table 6-3
File Specification Defaults

Field	Default
device name	At log in, the user's system device. May be changed subsequently by the SET command. The new default device must be mounted and the user must have access to it. Not to be defaulted when the file specified is to be written to or read from a record-oriented device (see Section 6.2.3)
ufd	At log in, the default UFD is equivalent to the user's UIC. May be changed subsequently by the SET DEFAULT command. A user must have access to any UFD selected as a default.
name	None
filetype	May be defaulted in the appropriate context. IAS has standard filetypes (see Table 6-2) that it uses as defaults in defined contexts.
version	For input specifications, the highest version number. For output specifications, the highest version increased by 1 or 1 if no previous version exists.

6.2.1.1 <u>Changing Default Values (The SET Command)</u> - The default device or UFD used in file specifications may be changed at any time by the SET command.

To change the default device:

```
     PDS> SET DEFAULT device-name
```

where device-name is the new default device.

To change the default UFD:

```
     PDS> SET DEFAULT ufd
```

where ufd is the new default UFD in the format [m,n] and m and n are octal numbers between 1 and 377. See Part 2 for a complete description of the SET command.

6.2.1.2 <u>Displaying Default Values (The SHOW Command)</u> - The current default values for the device field and UFD field can be displayed at an interactive terminal by using the SHOW command (see Part 2) as follows:

```
     PDS> SHOW DEFAULT
```

The system responds by displaying the user's default device and ufd.

6.2.2 Wild-cards

6.2.2.1 Input Files - The user may specify more than one file in a single input file specification by using an asterisk (*) convention called a wild-card. An asterisk may be placed in any field of a file specification except the device field.

The asterisk causes many commands to ignore the contents of the "wild" field and to select all the files that satisfy the remaining fields.

Examples:

DEL CATH.DAT;*	Delete all versions of the file named CATH.DAT stored on the default device and UFD.
DIR DK1:[200,200]*.LST	Display information about all the highest versions of files on DK1: in UFD [200,200] that are of type LST.
PRINT [30,4]*.MAC;*	Print all versions of the files on the default device in UFD [30,4] that are of type MAC.
DELETE [*,*]TONY.DAT;*	Delete all versions of the file named TONY.DAT in every directory on the user's default device.
COPY *[90,4]FORT.FOR;*	Illegal specification. The device field cannot be wild.

6.2.2.2 Output Files - When a wild-card (*) replaces a field in an output file specification, it instructs the system to replace the wild field with the corresponding field in the input file specification. The device field may not be wild.

Example:

```
PDS> COPY CATH.DAT

TO?  DK2:*.*
```

Copy the highest version of the file CATH.DAT from the default device to DK2:. If no version of CATH.DAT exists in the output file UFD, the version number of the output file is 1. If the output file UFD already contains one or more versions of CATH.DAT, the newly-copied CATH.DAT is given a version number one greater than the previously highest version.

Example:

```
PDS> COPY

FROM?  CATH.DAT

TO?  DK2:*.*;*
```

By placing a wild-card in the version field of the output file specification the user instructs the system to retain the same version number as the input file. The system returns an error message if the output file UFD contains a file with the same name, type and version number as the output file.

6.2.3 Valid File Specifications

The fields of a file specification that must be supplied, depend on the type of file being described. There are two types of file:

1. Retrievable files written to or stored on disks, DECtapes or magnetic tapes. These files are called named files because they have file names that the system can access.

2. Files that are read from or written to record-oriented devices (for example, a card reader or a line printer) or files held on unlabelled tapes. These files are called unnamed files.

The filename field of a named file must always be supplied; that is, the user must give an alphanumeric filename or a wild-card(*). Many commands have a default value for the filetype field. However, with any command that has no such default; the filetype field of a named file must always be supplied. The device, UFD and version fields may be omitted because they do have default values (see Section 6.2.1). The device field may also be replaced by a logical name (see Section 6.3.1).

The use of wild-cards in a file specification depends on the IAS command with which it is issued. Where it is relevant, the command descriptions in Part 2 describe restrictions on the use of wild-cards.

The specification of any unnamed file, a file read from or written to a record-oriented device, consists only of the device field, which may be a specific device or a logical name (see Section 6.3.1). If any other field is supplied, it is ignored by the system because UFDs, file names, filetypes and versions have significance only for named files. The device field may not be wild.

6.3 DEVICE MANAGEMENT

Before a batch or interactive user can access a device, the device must be available. In other words, the device must be attached to the system and, in the case of a removable volume, the volume must be physically loaded. Also, if the device is nonsharable, no-one else must be using it. For example, if all tape drives are already in use, the system cannot grant a new request for a tape drive.

If the conditions are such that a device is available, the user then gains access to the device by "allocating" it, that is, by issuing a command that requests the system's permission to use it (see Section 6.3.2). An exception to this procedure occurs when the user wants to access a system device.

6.3.1 System Devices

A system device is a device allocated to all users by the system manager. For example, the user's system disk, the line printer and the card reader are normally system devices.

A device such as a line printer cannot be shared by two users simultaneously, but many users may want to access it at the same time. The system manager may therefore choose to adopt a technique called spooling. In the case of a line printer, spooling causes all output written to the printer to be queued. The system then creates disk files of all line printer output, maintains a queue containing a list of these files and prints them one at a time.

Optionally, the printing of queued files can be deferred by the user via the command SET PRINTING DEFERRED (see 6.4.3.1). Deferred printing can be made the default at installations where, for example, the line printer is remote from the user.

6.3.2 Accessing a Device

In order to use a non-system device, three mechanisms are required:

1. A means of obtaining access to the device (the MOUNT and ALLOCATE commands).
2. A means of keeping commands, especially in batch mode, independent of a particular physical device (Logical Device Names).
3. A means of keeping the Input/Output statements in a program independent of a particular physical device (Logical Unit Numbers).

Access to a non-system device is obtained by issuing the ALLOCATE and/or the MOUNT command. Some devices, such as disk drives, are shareable. Thus a user may mount a disk even though it has already been mounted by another user. The volume is physically unloaded when the last user to access it dismounts it.

A user is granted exclusive access to non-shareable devices.

Note that access to any volume is subject to the normal protection restrictions (see Section 6.1.3).

6.3.2.1 Logical Device Names

IAS uses logical names to permit the commands written by a user to be independent of a particular physical device. If, for example, an installation has two tape drives called MT0: and MT1:, specifying MT0: in batch commands or indirect command files would prevent the user from using the other tape drive without changing the commands. The user may define a logical device name, TA:, for example, and use it in place of the corresponding physical device name in all subsequent commands.

Once an equivalence has been established between a logical device name and a physical device name, the logical device name may be used in any command. If a logical device name is the same as a physical device name, IAS assumes that the reference is to the logical device name.

Logical device names may be defined in ALLOCATE or MOUNT commands. A logical name has the syntax:

```
XX[nn]:
```

where XX represents two alphabetic characters and nn is an optional unit number, an octal number ranging from 0 to 77. If nn is omitted 0 is assumed.

6.3.2.2 Logical Units - All program Input/Output (I/O) is performed on logical units, which are identified by numbers (logical unit numbers or luns). Before a logical unit can be used for I/O, a physical device or file must be assigned to it. Since different devices or files may be assigned to the logical units on successive runs of a program, the program itself can be device-independent.

Users may assign logical units in three ways:

1. By using a LINK option during task build.
2. By issuing an ASSIGN command.
3. By establishing the assignment within the program before the file in question is accessed.

The LINK option and the ASSIGN command may be used to assign a physical or logical device to a logical unit. From within a program, however, the user may assign a named file to a logical unit. See one of the following manuals for further details:

1. The IAS Executive Reference Manual - Volume II
2. The appropriate IAS FORTRAN User's Guide
3. The PDP-11 COBOL Language Reference Manual
4. The BASIC-11 Language Reference Manual
5. The IAS/RSX-11 MACRO-11 Reference Manual

6.3.3 The MOUNT command

In order to access a file held on magnetic media, the volume on which it is held must be physically loaded and mounted. System devices that are already mounted when a user logs in are automatically mounted for him. For all other volumes, however, the user must issue a MOUNT command to make the device available and gain access to the volume residing on it.

Example:

```
PDS> MOUNT

DEVICE? DK2:

VOLUME-ID? TESTER
```

The command above mounts the volume labelled "TESTER" on DK2:. The user can now access any file on the mounted volume, as long as the file's protection code permits the attempted access.

Here the simple MOUNT command indicates that the volume is in IAS's Files-11 format. Volumes in Files-11 format have a volume-identification on the medium itself. This is set when the volume is initialized. The volume-identification is used when the volume is mounted or dismounted.

The unit number in the device specification may be omitted if the user does not know or care on which unit the volume is to be mounted. If the unit number has been omitted in batch mode, the user must then supply a logical name for the device; the logical name replaces the device name in subsequent file specifications. In interactive mode, the system displays a message giving the unit on which the volume was actually loaded.

Example:

```
    $MOUNT DK:  TESTER DRØ:
```

The user assigns the logical name DRØ: to the unknown unit. The logical name can now be used instead of the physical device name in subsequent commands.

Files-11 disks and DECtapes are shareable volumes which can be mounted and accessed by more than one user. Magnetic tape, however, can only be mounted and accessed by one user at a time.

The system considers any volume not in Files-11 format to be "foreign". A foreign volume can only be mounted by one user at a time and the system must be told that it is foreign. Volumes mounted "foreign" are normally referred to by some external label visible to the operator.

Example:

```
    PDS> MOUNT/FOREIGN

    DEVICE?  DTØ:

    VOLUME-ID?  TAPEA
```

The command qualifier /FOREIGN tells the system that TAPEA is not to be accessed as a Files-11 volume and prevents other users from mounting it. The operator mounts the volume, with external label TAPEA, on drive DTØ:

If the foreign volume is in DIGITAL's DOS or RT-11 format, file qualifiers to the COPY, DELETE and DIRECTORY commands allow the user to access files held on the volume. Otherwise, most PDS commands do not apply to foreign files.

See the specification of the MOUNT command in Part 2 for further details.

6.3.4 The DISMOUNT Command

When a user has finished accessing a volume, the DISMOUNT command should be issued in order to dismount the device and make it available for other users.

The DISMOUNT command automatically deallocates the device unless the user specifies the qualifier /KEEP. See the command specification in Part 2 for further details.

Examples:

```
1.   PDS> DISMOUNT

     DEVICE?  DK0:


2.   $DISMOUNT DT0:  TAPEA
```

The parameters to DISMOUNT are the device specification or logical name of the device to be dismounted and the volume identification.

6.3.5 The ALLOCATE Command

If a device is not a system device and it cannot be mounted, the ALLOCATE command must be used to access it.

Example:

```
PDS> ALLOCATE

RESOURCE?  DEVICE

DEVICE?  LP1:
```

The above example allocates a line printer to the user. No one else can use the printer until the user who allocated it issues a DEALLOCATE command (see Section 6.3.5).

The ALLOCATE command may also be used to obtain exclusive access to a shareable device.

Example:

```
$ALLOCATE DEVICE DK:  MC0:

DK3:  ALLOCATED

$MOUNT MC0 VOL1
```

In the above example, a batch user has allocated a DK type disk drive and assigned it the logical name MC0:. No one else is allowed to access that drive until is has been deallocated. PDS announces which physical device has been allocated for exclusive use to the user, here DK3.

Once a device has been allocated, several volumes may be mounted one after the other.

For example:

```
$ALLOCATE DEVICE DK:  DV1:

$MOUNT   DV1:   VOL1
```

```
    .

    .

$DISMOUNT/KEEP DV1:

$MOUNT DV1:   VOL2

    .

    .

    .

$DISMOUNT DV1:
```

In this example, the user obtains exclusive access to a disk drive via the ALLOCATE command. A volume labelled VOL1 is then mounted on the drive. When the user dismounts VOL1, the /KEEP qualifier retains the user's exclusive access to the disk. When VOL2 is dismounted, however, the disk is deallocated since the user does not specify /KEEP.

6.3.6 The DEALLOCATE Command

After issuing an ALLOCATE command to obtain exclusive use of a non-mountable device (a line printer or card reader, for example), a user must issue the DEALLOCATE command to free the device.

Example:

```
$ALLOCATE DEVICE LP1:
.
.
.
$DEALLOCATE DEVICE LP1:
```

The DISMOUNT command automatically deallocates an allocated mountable device unless the user specifies the /KEEP qualifier.

Example:

```
$ALLOCATE DEVICE DK:  MC0:
$MOUNT MC0:  CATH

     .

     .

     .

$DISMOUNT/KEEP MC0:

$DEALLOCATE DEVICE MC0:
```

6.3.7 The ASSIGN Command

The ASSIGN command is used to associate a logical or physical device with a logical unit. (See Section 6.3.2 for a definition of logical devices and logical units.)

Example:

```
     PDS> ASSIGN

     FILE?  LPØ:

     LUN?  6
```

This command assigns LPØ: to the logical unit 6. If a program writes to logical unit 6 via The FORTRAN statement WRITE (6..., for example, the results of the write will be printed on the line printer.

6.4 FILE MANAGEMENT

Section 6.5 describes the management of sequential files, that is, of the default file organization in IAS. For the extensions applying to Relative and Indexed files, compare Section 6.1.4 above and the commands APPEND, COPY, CREATE and MERGE in Part 2 of this manual.

6.4.1 Creating Files

6.4.1.1 User File Directories - To create a file on a volume, the volume must be mounted (see Section 6.3) and the user must have write access to a User File Directory (UFD) on the volume. A UFD is a file that contains details of all the files that have been created on that volume under the UFD identifier (i.e. [m,n] where m and n are octal numbers from 1 to 377).

Interactive users can issue the DIRECTORY command to display the contents of a User File Directory at the terminal. In batch mode, the directory information is sent to the user's output stream (TO).

Example:

```
     PDS> DIRECTORY


     DIRECTORY DBØ:[2ØØ,22]

     15-MAY-77 17:2Ø

     ADD.OBJ;1          2.              15-MAY-77 17:17
     ADD.FTN;1          1.              15-MAY-77 17:17
     ADD.TSK;1          32.         C   15-MAY-77 17:18

               TOTAL OF 35./35. BLOCKS IN 3. FILES
```

If no parameter is supplied, the system displays information about the user's current default UFD. However, by supplying one or more file specifications the user can interrogate other directories or specific files.

Example:

```
     PDS> DIRECTORY ADD.OBJ


     DIRECTORY DBØ:[2ØØ,22]

     15-MAY-77 17:2Ø

     ADD.OBJ;1           2.          15-MAY-77 17:17

             TOTAL OF 2./2. BLOCKS IN 1. FILE
```

To interrogate DOS or RT-11 files, modify the file specification with the /DOS or /RT11 file qualifier.

Example:

```
     PDS> DIRECTORY <altmode>

     FILE?  RTFILE.MAC/RT11
```

A User File Directory is like any other file; it has a protection code which determines who has access to it. A user may therefore create a file under any UFD to which he has write access.

6.4.1.2 The CREATE Command - Both batch and interactive users may create files by using the IAS command CREATE.

The interactive user types CREATE and supplies a file specification (no wild-cards allowed), optionally modified by the /PROTECTION qualifier. If the /PROTECTION qualifier is not specifically supplied, the new file is assigned the default file protection associated with the volume.

For example:

```
     PDS> CREATE

     FILE?  FORT.FTN/PRO:(OW:RWED SY:  GR:  WO:)
```

The system uses default values (see section 6.2.1) for the device, UFD and version fields.

Once the command string has been terminated, the user types input to the new file, line by line.

When terminated, each line is sent to the file exactly as it has been formatted at the terminal. The user then closes the file by typing CTRL/Z.

The batch user supplies the command name optionally modified by /DOLLARS and a file specification (no wild-cards allowed), optionally modified by the /PROTECTION qualifier. The qualifier /DOLLARS tells the system that the file will be closed by the $EOD command. Otherwise, any $ (i.e. batch) command terminates the file. Therefore, the /DOLLARS qualifier must be specified whenever a record in the file being created contains a $ in position 1. See Part 2 of this manual for other CREATE command qualifiers.

Examples:

1.
```
$CREATE/DOLLARS FORTRAN.FTN/PRO:(OW:RWED SY:  GR:  WO:)

        .
        .
        .

$EOD
```

2.
```
$CREATE DK2:[30,4]CALCULATE.MAC
```

6.4.1.3 Using the Editor to Create a Sequential File - Users can also create files by means of the EDIT command. See Chapter 7 for a description of the IAS text editors.

6.4.2 Manipulating Files

This section describes how to use various IAS commands to manipulate existing files in the following ways:

- To append one or more files to an output file
- To copy a file
- To rename an existing file
- To merge a file with an existing INDEXED or RELATIVE file.

6.4.2.1 The APPEND Command - The APPEND command may be used to add one or more files onto the end of an existing file.

Examples:

1.
```
PDS> APPEND (A.CBL, B.CBL)

TO? C.CBL
```

   Append files A.CBL and B.CBL to the end of the file C.CBL.

2.
```
$APPEND MYFILE.MAC YOURFILE.MAC
```

   Append MYFILE.MAC to the end of YOURFILE.MAC.

NOTE

A user must have extend access to a file before appending to it.

The user specifies the input file or files (enclosed in parentheses if more than one) first and then the output file.

Input files may be retrieved from a mounted volume, input from a record-oriented device (for example, a card reader) or typed in from an interactive terminal. When more than one input file is supplied, the system appends the files in the order in which they are specified.

If one of the files is to be input from the user's terminal (TI), the system transfers to the input file everything typed at the terminal after the command string until the user types CTRL/Z to close the file.

Example:

1. $APPEND (FILE1.MAC, FILE2.MAC), FILE3.MAC

   The system adds the input files FILE1.MAC and FILE2.MAC to the file FILE3.MAC.

2. <u>PDS></u> APPEND

   <u>FILE?</u> JUD.CBL

   <u>TO?</u> GRAVES.CBL

   The file JUD.CBL is appended to the output file GRAVES.CBL.

6.4.2.2 <u>The COPY Command</u> - The COPY command creates a duplicate of the contents of an input file in a specified output file. Optional command qualifiers allow the output file to be modified in various ways.

Examples:

1. <u>PDS></u> COPY

   <u>FROM?</u> MT2:FRED.MAC

   <u>TO?</u> DK2:JIM.MAC

2. $COPY MT2:FRED.MAC, DK2:JIM.MAC

The examples above copy the highest version of FRED.MAC on MT2: to DK2: and change the file name to JIM on DK2:. As well as copying from one device to another, the COPY command can be used to copy a file from one User File Directory to another.

Example:

1. <u>PDS></u> COPY [30,4]FRED.MAC

   <u>TO?</u> [100,100]FRED.MAC

This example copies the file FRED.MAC in [30,4] to UFD [100,100]. The filename remains unchanged.

Four of the possible command qualifiers are:

/ALLOCATION:n

/CONTIGUOUS

/OWN

/REPLACE

These are explained in detail in Part 2, but some examples of their use are shown below:

1. `$COPY/ALLOCATION:2Ø DK2:OLDFILE.DAT DKØ:OLDFILE.DAT`

   Copy OLDFILE.ONE from DK2: to DKØ: and make the output file 2Ø blocks long. The /ALLOCATION qualifier is useful for copying a contiguous file and changing its size.

2. PDS> COPY/CONTIGUOUS

   FROM? MT2:TU71.MAC   DK1:*.*

   Copy TU71.MAC from MT2: to DK1: and make the output file contiguous. The wild-cards (*) indicate that the fields of the output specification in which they occur take the corresponding field values of the input file specification (ie. the output file will also be named TU71.MAC.).

3. `$COPY/REPLACE MT1:SAME.OBJ;4 DK2:SAME.OBJ;4`

   The /REPLACE qualifier indicates that the output file overrides a file in the user's default UFD that has the same name, type and version number. That is, if a file called SAME.OBJ;4 already exists on DK2: in the default UFD, it is deleted and replaced by the new one copied from MT1:

There are two file qualifiers available with the COPY command, /RT11 and /DOS, that allow the user to copy files to or from an RT-11 or DOS formatted volume. The qualifier must modify the specification of the file currently in DIGITAL'S DOS or RT-11 format. DOS and RT-11 files cannot be renamed within IAS; therefore, the filename and filetype fields of the output file specification must always be wild.

Examples:

1. PDS> COPY

   FROM? DK2:FRED.DAT/RT11

   TO? *.*

   Copy the RT-11 file FRED.DAT from the foreign volume on DK2: to the user's default device and UFD. The /RT11 qualifier instructs the system to translate the RT-11 file into Files-11 format.

2. `$COPY TEST.MAC;8 DTØ:*.*/DOS`

   Copy the Files-11 file TEST.MAC;8 to a DOS-formatted foreign volume on DTØ:.

6.4.2.3 Renaming Files - The RENAME command may be used to change the name of a file. The examples below change the file name DEBUG.MAC;1 to RUN.MAC;1.

```
     1.  $RENAME DEBUG.MAC;1    RUN.MAC;1

     2.  PDS> RENAME

         OLD?  DEBUG.MAC;1

         NEW?  RUN.MAC;1
```

6.4.3 Listing Files

Sequential files may be listed on a line printer or at the user's terminal. One of the commands discussed below should be used; the choice is dependent on the kind of listing desired and whether the user is operating in interactive or batch mode.

6.4.3.1 Listing on the Line Printer - The PRINT command may be used to print files on the line printer. The system often queues all line printer output until all output previously submitted to the queue has been processed. The output files are normally printed in the order in which they were submitted to the queue.

The PRINT command is the simplest way to queue a file to the line printer. For example:

```
     1.  PDS> PRINT

         FILE?  FILE1.DAT, FILE2.DAT, FILE3.DAT

     2.  $PRINT LIST.MAP
```

The file or files to be printed are specified after the command.

The PRINT command provides the option to delete files after they have been printed. The user indicates this option by supplying the command qualifier /DELETE. For example:

```
     $PRINT/DELETE MYFILE.DAT
```

The actual printing can be deferred until later by the command SET PRINTING DEFERRED. Printing will then begin when the user logs out (by choice or timeout) or when SET PRINTING NODEFERRED is issued. NODEFERRED is the normal default for the system.

6.4.3.2 Printing on Varied Stationery - There can be up to seven distinct print queues. Each queue can be associated with a particular type of continuous stationery, for example, fan-fold, graph plotter paper, pay slips and so on. Within the system, these queues are refered to by number, n, say, with values from 0 to 6. Outside the system, the association of each value of n with a particular stationery is agreed from time to time according to the installation's needs.

n=0 is always the default queue. n=1 through 6 can be used only via the PRINT or the QUEUE command or the PRINT$ MACRO directive. The PRINT command directs the file to the CL device, that is, the device to which CL was redirected at IAS start up. QUEUE can send a file to any spooled output device, with CL as the default output device.

When output spooling is enabled, the system firsts prints all the n=0 queue on the CL device. If printing is queued with other values of n, the system informs the operator via the system console whenever a change of stationerv is required.

6.4.3.3 <u>Listing Files at an Interactive Terminal</u> - The TYPE command causes one or more specified files to be printed at the user's interactive terminal.

Examples:

1. `PDS> TYPE`

   `FILE?  FIRST.MAC, SECOND.MAC`

2. `PDS> TYPE TYPE.CBL`

6.4.3.4 <u>The DUMP Facility</u> - The DUMP command lists a specified file on the user's terminal (TO) or sends the listing to a specified output file. Command qualifiers modify the form of the listing. For example, the user may specify that the file be dumped in ASCII mode. The DUMP facility is useful for debugging programs and for displaying nonprintable characters in ASCII or octal format. See the full specification of DUMP in Part 2 for all the available options.

Examples:

1. `PDS> DUMP/ASCII`

   `FILE?  DUMP.CBL`

   List the file DUMP.CBL in ASCII format on the user's terminal.

2. `$DUMP/BYTE/OUTPUT:DK2:DISKFILE.DAT    OBJECT.DAT`

   Send a listing of the file OBJECT.DAT in byte octal format to a file named DISKFILE.DAT on DK2:

3. `PDS> DUMP/OUT:LP0:  FILE.DAT`

   List the file FILE.DAT in word octal format (the default) on the line printer.

6.4.4 <u>Deleting Files</u>

The DELETE command deletes files held on Files-11 disks or DECtapes, or DIGITAL's RT-11 or DOS files held on foreign disks or DECtapes.

Specifications of DOS or RT-11 files must be modified by a file qualifier, either /DOS or /RT11 as appropriate.

Wild-cards(*) (see Section 6.2.2) are allowed in the file specification. If the version field is omitted, the command qualifier

/KEEP:n may be supplied to preserve the highest n versions of the file or files specified.

Examples:

1. `PDS> DELETE/KEEP:2`

   `FILE? MATRIX.DAT`

   Delete all but the last 2 versions of the file MATRIX.DAT

2. `$DELETE   ROW.OBJ;4   COLUMN.MAC;4   PEEK.*;*`

   Delete all files named PEEK and the fourth version of the files ROW.OBJ and COLUMN.MAC.

3. `PDS> DELETE DK2:DOSFILE.DAT/DOS`

   Delete the file DK2:DOSFILE.DAT, which is in DIGITAL's DOS format.

The PRINT command modified by the /DELETE qualifier can be used to delete files that have been submitted to the line printer. See Section 6.4.3.1.

6.4.5 Summary of File Handling Commands

Command	Function
ALLOCATE	Allocate a specified device to the user.
APPEND	Add one or more files to the end of a specified sequential file.
ASSIGN	Assign a device to a logical unit.
COMPARE	Compare two files with one another and produce a summary of the differences found.
COPY	Copy an input file to a specified output file.
CREATE	Create a file as specified. File contents to be input from an interactive terminal, or, in batch, to follow immediately after the $CREATE command.
DEALLOCATE	Deallocate a specified device.
DEASSIGN	Deassign a device from a logical unit.
DELETE	Delete specified Files-11, DIGITAL's DOS or RT-11 formatted files.
DISMOUNT	Dismount a specified volume.
DUMP	List the contents of a file.
EDIT	Edit an existing file or create a new file.
INITIALIZE	Initialize a foreign (DOS or RT11) volume.

MERGE	Merge a file with an existing indexed or relative file.
MOUNT	Make a volume available to the user.
PRINT	Print one or more files on the line printer.
RENAME	Change the name of an existing file.
SET PROTECTION	Assign a specified protection code to a file.
SORT	Sort files into a specified sequence.
TYPE	List a file at the user's interactive terminal.

CHAPTER 7

IAS TEXT EDITORS

This chapter provides the user with the basic information needed to run either of the two IAS editors:

- The Text Editor (EDI), primarily for interactive use, and

- The Source Language Input Program and Editor (SLIPER), a batch-oriented editor.

The _IAS Editing Utilities Reference Manual_ contains a complete description of both editors.

7.1 THE TEXT EDITOR

The EDIT command automatically invokes the Text Editor, also known as EDI, unless the qualifier /SLIPER has been specified. EDI is an interactive context-editing program that uses editor commands to create and modify source programs and other files containing ASCII data. The specification of the EDIT Command in Part 2 contains a complete list of editor commands. This section introduces some basic editing concepts and describes a useful subset of commands.

Editor commands, as in PDS, describe the action to be performed. Each command consists of a command name followed by a single parameter. Most command names can be abbreviated to 1, 2 or 3 letters. Some command names, however, are themselves abbreviations of their function and cannot be abbreviated further. For example, NP which stands for Next Print, has no alternative form.

7.1.1 Editing Modes

EDI operates in two modes: input mode and edit mode. In input mode, EDI considers all lines entered at the terminal to be input to the file. This mode is used to create a file and to insert lines of text into an existing file. In edit mode, EDI treats lines entered at the terminal as editor commands intended to modify or manipulate existing text.

7.1.2 Input Mode

7.1.2.1. Creating a New File - if the user specifies a non-existent file with the EDIT Command, EDI automatically creates a new file and enters input mode. The specification of the file must include filetype. For example:

```
PDS> EDIT NEWFILE.DAT
[CREATING NEW FILE]
INPUT
```

The user then begins to enter text on the next line. All characters typed are written to the file. The function and CTRL characters are used to format the lines of text (see Chapter 3).

To enter a blank line into the text, type one or more spaces at the beginning of a new line, followed by carriage return.

7.1.2.2 The INSERT Command - If EDI is already operating in edit mode (see 7.1.3, 7.1.3.1), the editor command INSERT, immediately followed by carriage return, changes the operating mode to input.

7.1.2.3 Changing to Edit Mode - To switch from input to edit mode, type carriage return as the first character in a line. EDI responds by displaying an asterisk (*) on the next line. The asterisk is the EDI prompt for editor commands.

7.1.2.4 Closing a File - To close a file when in input mode, switch from input to edit mode. Wait for the asterisk prompt, then type EXIT (see Section 7.1.4.3).

7.1.3 Edit Mode

The asterisk (*) prompt indicates that EDI is operating in edit mode, and is therefore only accepting editor commands.

7.1.3.1 Editing an Existing File - To edit an existing file, supply the specification of the file with the EDIT command. The specification must include a filetype. If the version number is omitted, EDI selects the highest version of the file. EDI then retrieves the input file and prompts for an editor command. For example:

```
PDS> EDIT
FILE? OLDFILE.DAT
[n LINES READ IN]      where 'n' is the block size or file size,
                        whichever is the smaller.
[PAGE    1]
*
```

7.1.3.2 Block Editing - By default, EDI accesses a file in 80-line blocks, called pages. (This chapter discusses only this method of access to the file; the alternative method, called line-by-line mode, is described in the IAS Editing Utilities Reference Manual.) The editor command SIZE may be used to change the number of lines per page (see the specification of the EDIT command in Part 2).

7.1.3.3 The Line Pointer - EDI is a context editor; it locates the line to be edited by means of text contained within the line, rather than by sequence numbers, as does the batch editor SLIPER, described in Section 7.2. EDI uses a line pointer to indicate the current line to be edited.

When edit mode is first entered, the line pointer points to a line immediately preceding the first line of text in the file. The user then moves the line pointer by searching for a particular piece of text or by using commands that reposition the pointer.

For example:

```
PDS> EDIT NEWFILE.DAT
[CREATING NEW FILE]
INPUT
THIS IS LINE 1 ENTERED
HERE IS LINE 2
LINE 3
LINE 4 WHICH IS ALSO THE LAST LINE
<CR>
*TOF
[00005 LINES READ IN]
[PAGE     1]
*LOCATE LINE 1
THIS IS LINE 1 ENTERED
*NEXT
*PRINT
HERE IS LINE 2
*LOCATE ALSO
LINE 4 WHICH IS ALSO THE LAST LINE
*LOCATE ENTERED
[*EOB*]
```

In this example, the user has created a file consisting of 4 lines. When the prompt for editor commands (*) appears, the user issues the TOF (Top of File) command to move the line pointer to the top of the file. "Top" means the line immediately preceding the first line of text. The user then types "LOCATE LINE 1" to find the first line that contains the character string "LINE 1". EDI moves the pointer to that line and prints it. The LOCATE command always searches down the file beginning at the line immediately following the current line.

The NEXT command is used to advance the line pointer to the next line, which is "HERE IS LINE 2". The PRINT command then causes EDI to display the new current line without moving the pointer. "LOCATE ALSO" causes the line pointer to be moved to the fourth line, which contains the word "ALSO". This line is automatically printed.

The command "LOCATE ENTERED" causes the editor to print "[*EOB*]. "EOB" is the abbreviation for End of Buffer. Since the line pointer moves only down the text when searching for character strings, it encounters the end of the buffer without finding the string "ENTERED." The TOF command could then be used to reposition the line pointer at the beginning of the text.

7.1.4 Editor Commands

This section describes a useful subset of EDI commands. The complete set is listed in the specification of EDIT in Part 2. The IAS Editing Utilities Reference Manual specifies all the commands in detail.

The subset of commands is described in alphabetical order. All commands are separated from their parameters by one or more spaces. Brackets ([and]) indicate that the enclosed value is optional.

Note that the function keys carriage return (CR or RETURN) and ALTmode (ALT and ESC) can be used as editor commands. See the description of the NP Command, Section 7.1.4.8. CTRL/Z may also be used to close the editing session and return control to PDS; but it is advisable to use the editor command EXIT for this purpose.

The subset includes:

Commands	Sections
CHANGE	(Section 7.1.4.1)
DELETE	(Section 7.1.4.2)
EXIT	(Section 7.1.4.3)
FIND	(Section 7.1.4.4)
INSERT	(Section 7.1.4.5)
LOCATE	(Section 7.1.4.6)
NEXT	(Section 7.1.4.7)
NP	(Section 7.1.4.8)
PRINT	(Section 7.1.4.9)
PLOCATE	(Section 7.1.4.1Ø)
RENEW	(Section 7.1.4.11)
RETYPE	(Section 7.1.4.12)
TOF	(Section 7.1.4.13)

7.1.4.1 The CHANGE Command

Format

```
     [n]CHANGE /string-1/string-2[/]
```

where string is a character string. The slashes (/) delimit each string, and are therefore called delimiters. The delimiters may be any matching characters that do not appear in either string. The first character following the command is considered to be the first delimiter. The closing delimiter is optional.

n is a positive integer.

The command name may be abbreviated to one or more letters.

Function

This command searches for string-1 in the current line and, if found, replaces it with string-2.

If string-1 is given but EDI cannot locate the string in the current line, EDI prints "NO MATCH" and returns an * prompt. The command can be reentered using the correct string construct.

If string-1 is null (not given), string-2 is inserted at the beginning of the line. If string-2 is null, string-1 is deleted from the current line.

The search for string-1 begins at the beginning of the current line and proceeds across the line until a match is found. If string-1 occurs more than once on the current line, only the first occurrence is changed.

A different command is needed to change every occurrence in the line from string-1 to string-2. This is LC, standing for Line Change. See example 3 below.

A numeric value n preceding the command CHANGE causes the command to be obeyed n times. For each repetition, the entire line is rescanned beginning at the first character in the line. This allows the user to generate a string of n characters as shown in example 4 below.

If no match occurs, a NO MATCH message is displayed.

The Line Pointer

The CHANGE command does not change the position of the line pointer.

Examples

1. The current line reads "333". The following command changes it to "C33":

   ```
   *C/3/C/
   ```

2. The current line reads "DIAGNOSIS". The following command changes it to read "DIAGNOSTICS":

   ```
   *CHA "IS"TICS"
   ```

3. The current line contains "ABACAD". The following command changes it to read "XBXCXD":

   ```
   *LC/A/X
   ```

4. The current line contains "A;B;C;D". The following command changes it to read "A;;;;;B;C;D":

   ```
   *4C/;/;;
   ```

7.1.4.2 The DELETE Command

Format

```
     DELETE [n]
```

where n is a positive or negative integer.

The command name may be abbreviated to one or more letters.

Function

This command causes lines of text to be deleted in the following manner:

1. If n is positive, the current line and n-1 lines following the current line are deleted. The line-pointer advances to the line following the last deleted line.

2. If n is negative, the current line is not deleted, but the specified number of lines that precede it are deleted. The line pointer remains unchanged.

3. If n is omitted, the current line is deleted and the line pointer advances to the next line.

The Line Pointer

See items 1, 2 and 3 in the Function section above for the command's effect on the line pointer. To print out the line pointed to after the deletion type DP (Delete and Print) in place of DELETE.

Examples

To delete the previous five lines in the block buffer, type the following command:

```
        *D -5
```

To delete three lines and print the resulting line pointed to by the line pointer, type

```
     *DP 3
```

7.1.4.3 The EXIT Command

Format

```
     EXIT
```

The command name may be abbreviated to two or more letters.

Function

This command transfers all remaining lines in the block buffer and input file (in that order) into the output file, closes the file and causes EDI to exit. The system then prompts for PDS commands.

Example:

```
     *EX
     [EXIT]


     PDS>
```

7.1.4.4 The FIND Command

Format

```
     [n]FIND [string]
```

where n is a positive integer and string is a character string that begins in the first position of a line. The command name may be abbreviated to one or more letters.

Function

This command searches the block, beginning at the line following the current line, for string, which must begin in column one of the lines searched. If string is not specified, the line pointer simply advances one line. If n is given, EDI searches for the nth occurrence of string and positions the line pointer at the line that contains it.

FIND is useful for locating FORTRAN statement numbers and MACRO-11 statement labels.

The Line Pointer

If string is not given, the line pointer advances one line.

If string is given, the line pointer moves to the first or nth line containing string.

Example

```
     *F LOOK
     LOOK AT THE FIRST CHARACTER IN THE LINE
```

The above command causes EDI to search the block for a line beginning with LOOK and to print the line when it is found.

7.1.4.5 The INSERT Command

Format

```
     INSERT [string]
```

where string is a character string.

The command name may be abbreviated to one or more letters.

Function

This command inserts string immediately following the current line. If string is omitted, EDI enters input mode.

The Line Pointer

The line pointer moves to the line in which string is inserted, that is, the line following the current line.

Example

*I TEXT INSERT IN EDIT MODE	Inserts a line of text immediately after the current line.
*F ABC	Finds a line beginning with ABC.
ABC IS THE START OF THE ALPHABET	This is the line found.
*I <CR>	An I followed by a carriage
TEXT INSERT 1 IN INPUT MODE	return causes EDI to switch
TEXT INSERT 2 IN INPUT MODE	to the input mode and a
ETC.	series of new lines can be input following the current line.
<CR>	An extra <CR> closes the input and causes a return to EDIT mode.
*	Prompt for EDIT mode.

7.1.4.6 The LOCATE Command

Format

```
     [n]LOCATE [string]
```

where n is a positive integer and string is a character string.

The command name may be abbreviated to 1 or more letters.

Function

This command causes a search of the buffer beginning at the line following the current line for string, which may occur anywhere in the line sought. If string is not specified, the line following the current line is considered a match. A numeric value n preceding the command results in locating the nth occurrence of string. EDI then prints the located line.

The Line Pointer

EDI moves the line pointer to the line containing string or the nth occurrence of string.

Example

See Section 7.1.3.3.

7.1.4.7 The NEXT Command

Format

```
     NEXT [n]
```

where n is a positive or negative integer.

The command name may be abbreviated to one or more letters.

Function

If n is omitted, this command causes the line pointer to advance one line.

If n is supplied, the line pointer moves forward n lines if n is positive, or back n lines if n is negative.

Example

The following command moves the current line pointer back five lines:

```
     *N -5
```

7.1.4.8 The NP (Next Print) Command

Format

```
     NP [n]
```

where n is a positive or negative integer.

The command cannot be abbreviated.

Function

This command has the same function as the NEXT command (see Section 7.1.4.7) except that it prints out the new current line.

Note that pressing carriage return (CR or RETURN) performs the same function as NP 1, and pressing ALTmode (ALT or ESC) performs the same function as NP -1.

Example

The following four lines are contained in the file and the line pointer is at the first line.

```
LINE 1 OF THE FILE
LINE 2 OF THE FILE
LINE 3 OF THE FILE
LINE 4 OF THE FILE
```

If the following command is issued, EDI would return the following printout

```
*NP 2
LINE 3 OF THE FILE
```

7.1.4.9 The PRINT Command

Format

```
PRINT [n]
```

where n is a positive integer.

The command name may be abbreviated to one or more letters.

Function

This command prints out the current line and the next n-1 lines on the terminal. If n is omitted, the command prints the current line.

The Line Pointer

The line pointer is positioned at the last line printed if n is given. If n is omitted, the line pointer does not move.

Example:

```
*P3              Prints out the current line then the next two
                 lines.
LINE 1
LINE 2
LINE 3
*P               Prints the current line only (the last line
                 printed by the previous command).
LINE 3
```

7.1.4.10 The PLOCATE (Page Locate) Command

Format

```
[n]PLOCATE [string]
```

where n is a positive integer and string is a character string.

The command name may be abbreviated to two or more letters.

Function

This command searches for string in the current block and successive blocks, starting from the line following the current line. String may be positioned anywhere in the line in which it occurs. If n is specified, EDI searches for the nth occurrence of string. The line containing string is then printed.

If string is omitted, the line pointer advances one line.

The Line Pointer

The line pointer advances to the line containing string (or the nth occurrence of string) or to the line following the current line if string is omitted.

Example

The following command locates the line "HAPPY DAYS ARE HERE AGAIN", which occurs somewhere in the file ahead of the current line.

```
*PL PPY
HAPPY DAYS ARE HERE AGAIN
```

7.1.4.11 The RENEW Command

Format

```
RENEW [n]
```

where n is a positive integer.

The command name may be abbreviated to three or more letters.

Function

If n is omitted, the command writes the current block into the output file and reads a new block into the buffer. If n is specified, EDI reads n-1 blocks into the buffer and then writes them to the output file. The nth block is then read into the buffer and the line pointer positioned at the top of it.

Example

```
*RENEW 10
```

In this example, ten consecutive blocks are transferred from the input file to the block buffer. Only nine blocks, however, are transferred to the output file. The current line pointer is pointing to the first line in the tenth block which is currently in the block buffer.

7.1.4.12 The RETYPE Command

Format

```
RETYPE [string]
```

where string is a character string.

The command name may be abbreviated to one or more letters.

Function

This command replaces the current line with string. If string is omitted, the command deletes the current line.

The Line Pointer

The line pointer does not move.

Example

```
*RETY THIS IS A NEW LINE
```

In this example, the string "THIS IS A NEW LINE" replaces the current line.

7.1.4.13 The TOF (Top Of File) Command

Format

```
TOF
```

The command cannot be abbreviated.

Function

This command returns the line pointer to the top of the input file and saves all blocks (pages) previously edited. The "top" of the file is the line that immediately precedes the first line of text in the file.

Example

```
*TOF
```

This command causes the previously edited pages to be written into the output file. The line pointer then moves back to the top of the file.

7.1.5 Error Messages

Refer to the IAS Editing Utilities Reference Manual for a list of EDI error messages and recommended responses.

7.2 BATCH EDITING

The Source Language Input Program and Editor (SLIPER) is a batch-oriented editing program used to create and maintain source language files on disk. It permits the user to:

1. Edit an existing source file. Commands are provided to:

   a. Delete

   b. Replace

   c. Insert

2. Obtain line number listings of files.

SLIPER accepts input from

1. The input stream

2. Any Files-11 volume, i.e. a disk or DECtape in IAS format (see Chapter 6, Section 6.1.1)

Before starting SLIPER, the user should be aware of the following:

1. Lines can be located either by line number or by character strings within the line. A current listing must therefore be at hand, containing the line numbers if these are to be used.

2. The batch editor does not accept input lines greater than 80 ASCII characters in length. If more than 80 characters are specified, an error is declared.

3. Line numbers to which the edit commands refer must be in ascending sequence throughout the SLIPER file. Form feeds and page directives are treated as part of the text.

The PDS command COMPARE can compare two files line by line and generate the SLIPER input needed to convert one file to the other.

7.2.1 Invoking SLIPER

To invoke SLIPER the user must issue the EDIT command modified by the command qualifier /SLIPER. For example:

```
$EDIT/SLIPER OLDFILE.MAC
```

Further EDIT command qualifiers applicable only to SLIPER determine the format of the output files.

Table 7-1 lists the SLIPER qualifiers and their effects.

Table 7-1
SLIPER Qualifiers

Qualifier	Description	Default
/OUTPUT[:filespec]	Produce an output file. Unless filespec is specified, the file is given the same name as the input file, with a version number increased by 1.	/OUTPUT
/NOOUTPUT	Do not produce an output file.	
/LIST[:filespec]	If /OUTPUT has been specified print a listing of the output file on the line printer.	/LIST
	If /NOOUTPUT has been specified, print a listing of the input file on the line printer.	
	If filespec has been specified, name and store a listing file accordingly.	
/AUDIT	Enable the editing audit trail, which indicates in the output file the changes made during the most recent editing session.	/AUDIT
/AUDIT:(parameters)		
POSITION:n	Define the position n on the output file line where the audit trail is to be placed. The value n will be rounded to the nearest tab stop. (Default position is column 8Ø)	
SIZE:m	Define the maximum length of the audit trali string. (Default size is 8)	
	NOTE: Values for the position and size of the audit trail must be chosen such that the line does not exceed 88 characters.	
/NOAUDIT	Disable the editing audit trail.	
/BLANK	Insert blanks at the end of the text line (rather than tabs) to right-justify the audit trail text.	/BLANK
/NOBLANK	Do not insert blanks at the end of the text line.	
/DOUBLE	Produce a double-spaced listing file.	/NODOUBLE
/NODOUBLE	Produce a single-spaced listing file.	

7.2.1.1 Obtaining a Listing - Note that to produce a listing of the file to be edited, the user must specify the /NOOUTPUT qualifier. For example:

```
PDS> EDIT/SLI/NOOUTPUT/LIST CHARLES.MAC
```

The command above prints a listing of CHARLES.MAC on the line printer. In batch, the default is /LIST, but interactive users must specify that qualifier to obtain a listing. The listing provides the line numbers to be used in subsequent editing of CHARLES.MAC.

7.2.2 SLIPER Output Files

When a file is edited, SLIPER produces an output file on disk under the name specified by the user. If the /AUDIT qualifier is specified (default condition), the file contains an audit trail, indicating changes effected by the editing session.

Each line that has been inserted during the last editing session is flagged by appending the characters ;**NEW** to the line. The user may reset **NEW** to a flag of his own choice.

The line following the inserted line(s) may be flagged by the characters ;**-n, where n is a decimal value equal to the number of lines that were deleted from the old file. For example:

```
;THIS IS A NEW LINE ADDED TO THE FILE ;**NEW**
;THIS IS THE NEXT LINE                ;**-1
```

indicates that the new line has simply replaced one of the old lines; that is, the edit command looked like:

```
;THIS IS A NEW LINE ADDED TO THE FILE
-m, m
```

where m is the number of the line that was replaced. There may also be entries of the following kind:

```
;THIS LINE IS A REPLACEMENT ;**NEW**
;NEXT OLD LINE              ;**-16
```

indicating that a new line has been inserted, but 16 lines have been deleted immediately preceding the next old line.

Lines may also be flagged with the characters ;**N, with no preceding new lines, to indicate that lines have been deleted without being replaced.

If /AUDIT has been specified, the current flags are stripped before the updated file is output; thus, the flags are reliable indicators of the most recent update of the file.

7.2.3 SLIPER Edit Commands

7.2.3.1 SLIPER Editing Command Formats

For an insertion:

```
-location1[,/audit-trail][;]
```

Insert text following the line in the input file given by location1.

For a deletion:

```
-location1,location2[,/audit-trail][;]
```

Delete line(s) given by location1 through location2.

where:

location1 and location2 are

n	n is a line number (decimal)

or

/string/[+n]	string is an ASCII string and may occur anywhere in the line to be located. Within string, three periods ... can be used to represent omitted characters. +n, if used, advances the location a further n (decimal) lines.

or

.[+n]	current position [advanced in lines].
audit-trail	is an ASCII string to be appended to each new line of text if /AUDIT is in force. Default (if /AUDIT is in force) is the immediately previous setting of audit-trail. Initial setting: ;**NEW**
;	remainder of line following ; is a comment.

7.2.3.2 Location by Line Numbers

Following the initial invocation of SLIPER (7.2.1), the user enters text lines, or deletes or corrects lines in the original source file. Text that is to be inserted at the beginning of the file is entered immediately following the initial command line. To correct or replace one or more lines, or to insert text in the middle or at the end of the file, the user must first specify an edit command in line position 1, followed by a decimal value that refers to a line in the input file. For example:

```
-9
```

The minus sign and line number may appear as the only element on the line, or they may be followed by a comma and a second line number, as:

```
-9,12
```

or

```
-9,9
```

SLIPER interprets the user's purpose by examining the edit command. When a single line number is specified (e.g. -9 alone), SLIPER interprets the user's purpose to be the insertion of new text lines into the source file. The line number indicates that the new text is to be inserted following the specified line (in the first example, new text would be placed in the file following line 9).

When the user provides an edit command in the second format (-9,-12), SLIPER deletes all text lines from line 9 through line 12, inclusively. The user can follow the edit command with lines of text, which will be inserted into the file in the location previously occupied by the deleted lines (that is, the first new line is the new line 9).

The edit command (-9,9) indicates that SLIPER is to delete line 9. If a text line (or lines) follows. It replaces the deleted line.

NOTE

> Line numbers must always be specified in ascending sequence. Thus, -9,8 is illegal, and an error message is printed. It is also illegal to refer to a line number lower than a line number that was referred to in a prior edit command.

In place of n the user can specify a single period to mean 'the current line'. '.+n' means 'the current line advanced n further lines'.

7.2.3.3 Location by Character String - Instead of a line number n the user can at any time specify a character string occurring within the line. SLIPER locates the next line containing a matching string. The string itself cannot contain a slash, because slashes are used to delimit the string in the command. The user can also specify an advance of a number of lines from the first matching string found.

Instead of a complete string the user can specify one or more characters at its beginning and at its end. Three periods ... must be used to represent the omitted characters.

7.2.3.4 <u>SLIPER Edit Control Characters</u> - SLIPER recognises four characters as edit control characters when they appear in line position 1:

The minus sign (-)

The "less than" sign (<)

The slash (/)

The "at" sign (@)

Table 7-2 describes their use as edit control characters.

Table 7-2
SLIPER Edit Control Characters

Character	Function
-(minus)	Indicates that an editing function is to be performed, with reference to the lines specified by number or character string. -n Insert text following line n. -n,n Delete line n. -n,m Delete lines n through m inclusively (m must be greater than n). For the alternatives to n or n,m see 7.2.3.1.
/(slash)	The slash is placed in the first position of a line to indicate that the editing of a file is completed.
@(at)	The @ character is put in the first location of a line to indicate that SLIPER is to seek input from an indirect file. The user must specify the indirect file immediately after the @ sign; for example: @DK2:DKSFIL.CMD instructs SLIPER to read input from the file DKSFIL.CMD on physical device unit DK2:. Indirect files are more fully described in Section 7.2.4. Unless otherwise specified, the file extension defaults to .CMD.
<(less than)	The < character is used when entering a line that begins with one of the special edit control characters. It causes the line to be shifted one character to the left, with the result that the < is deleted, and the desired control character is entered into the file as the first character on the line.

7.2.4 Indirect Files

Indirect files can be used to contain both editing commands and correction lines to be inserted into the file being edited. (See Chapter 8, Section 8.2.)

7.2.5 SLIPER Editing Examples

The following examples show the various editing functions that SLIPER can perform, and the command formats used.

EXAMPLE A

```
$EDIT/SLI JONES.MAC

-23,23

; R1=SIZE OF BLOCK TO ALLOCATE IN BYTES

-33

     MOV    $FRHD,R2  ;GET ADDRESS OF FREE POOL HEADER

-36,36

-39,39

     ASR     R1       ;CONVERT TO WORDS

/
```

This example performs the following editing functions:

- Line 23 is replaced by a corrected version (i.e.; R1 = SIZE OF BLOCK TO ALLOCATE IN BYTES.);
- A new line is inserted after line 33;
- Line 36 is deleted (and not replaced);
- Line 39 is replaced by a corrected version (i.e., ASR R1 ;CONVERT TO WORDS),

EXAMPLE B

```
$EDI/SLI CATHS.MAC

-55,55

       BCS    60$     ;IF CS YES

-107,107

       CALL    $ERMSG ;OUTPUT ERROR MESSAGE

/
```

Example B performs the following editing functions:

- Line 55 is replaced by a corrected line;
- Line 107 is replaced by a corrected line.

EXAMPLE C

```
$EDI/SLIP/OUTPUT:CHAS.MAC   CATHS.MAC

-15,16
CNTRL:   .BYTE     '9,'Ø

-33,35

$CDTD::  MOVB      #'9,CNTRL    ;SET DECIMAL LIMIT

-38,38

COTB::   MOVB      #'Ø,CNTRL    ;SET OCTAL LIMIT

-43,45


         CMPB      #' ,R5       ;BLANK?

         BEQ       1$           ;IF EQUAL YES

         CMPB      #HT,R5       ;HT?

         BEQ       1$           ;IF EQUAL YES

-/3$:/,.+3

3$:      MOV       R5,R2        ;SET TERMINAL CHARACTER

/
```

Example C performs the following editing functions:

- Lines 15 and 16 are deleted and replaced by a corrected line;
- Lines 33 through 35 are deleted and replaced by the line starting with $CDTD;
- Line 38 is replaced;
- a line including the string "3$:" and the following 3 lines are replaced by a single line;
- Line beginning with 3$: is inserted.
- The output file is created under the name CHAS.MAC.

CHAPTER 8

INTRODUCTION TO PROGRAM CONTROL

IAS supports several programming languages, including BASIC, COBOL, FORTRAN, CORAL and MACRO-11. MACRO-11 is a standard feature of IAS; the other language translators are optional. This chapter is an introduction to some language-independent aspects of running programs under IAS. The next five chapters, one on each language, describe how to use IAS commands to transform source programs into executing programs or tasks.

8.1 PROCESSING MODES

Whether it is better to operate in batch or in interactive mode depends on the nature of the programmer's job and the requirements of the installation. Interactive mode is convenient for complicated editing of source programs, for instance, or the execution of programs that require small amounts of input data. On the other hand, batch processing is usually the best mode for processing large amounts of data, for example, a payroll or accounts receivable.

8.2 INDIRECT FILES

An indirect file is a sequential file containing command input. For example, rather than repeatedly typing commonly used command sequences, the user can type the sequence once and store it in a file. To execute the sequence, the user issues an "at" sign (@) followed by the file specification instead of the first command in the sequence. The indirect file may be invoked from any position within the command string, but any characters that follow the indirect file specification are ignored. The system then retrieves the indirect file and executes the commands contained therein.

Example:

```
PDS> EDIT FILE.CMD

[CREATING NEW FILE]

INPUT

FORTRAN/OBJECT/LIST:CPROG   CPROG

LINK CPROG

RUN CPROG
```

```
          <CR>

          *EXIT

          PDS>

          .
          .
          .

          PDS> @FILE
```

The indirect file called FILE.CMD, created by means of the Line Text Editor, contains commands to compile, link and run the source program CPROG.FTN.

These commands are executed when the user invokes the file by typing @FILE in response to the PDS prompt. CMD is the default filetype for indirect files.

In a batch context, the same command sequence could be created and invoked in the following manner:

```
          $CREATE/DOLLARS FILE.CMD

          $FORTRAN/OBJECT/LIST:CPROG CPROG

          $LINK CPROG

          $RUN CPROG

          $EOD
          .
          .
          .

          @FILE
```

Note that the $CREATE command string must include the qualifier /DOLLARS, so that the system recognizes the following text as input and not as further batch commands to be processed. The $EOD command terminates the file to be created.

The command file may be invoked subsequently by the command line @FILE. No dollar sign ($) is needed.

Both batch and interactive users may invoke indirect files on up to three levels. An indirect file can itself invoke another indirect file; the second file may invoke a third; but the third file may not invoke a fourth indirect file.

See Chapter 6, Section 6.4.1 for a description of file creation.

8.3 USER LIBRARIES

The IAS command LIBRARIAN allows users to create and maintain their own libraries of commonly-used macros (macro libraries) and routines (object module libraries).

8.3.1 Macro Libraries

MACRO-11 macros may be held in source (text) form in a macro library. Each macro is identified by its macro name. To use one or more macros contained in a macro library file, the programmer must supply the library file specification, modified by the qualifier /LIBRARY, in the list of input files to the MACRO command. (See the description of the MACRO command in Part 2.) The macro library must be specified before the module that calls it.

8.3.2 Object Module Libraries

Commonly-used routines are stored in object (that is, compiled or assembled) code which the user can then incorporate in a task. The object code routines are called object modules; the files in which they are held are called object module libraries.

A programmer who invokes a library object module must ensure that the module is linked at task build time. The Task Builder automatically searches all system libraries; but it only searches user-written libraries that have been explicitly specified in the LINK command (see Part 2, LINK, the file-qualifier /LIBRARY and LINK/OPTIONS).

The IAS Task Builder Reference Manual describes object module libraries in detail.

The specification of the LIBRARIAN command in Part 2 describes how to create and maintain the libraries.

8.4 CREATING SOURCE FILES

Either the CREATE command or the EDIT command may be used to create source files. The EDIT command has the advantage that it allows the user immediate access to all its editing facilities. To correct errors made while using the CREATE command, however, the user must rely on keyboard facilities or close the file and then issue the EDIT command.

8.4.1 The CREATE Command

To create a source file with the CREATE command, the user must do one of two things:

1. In batch mode, issue the command $CREATE, optionally modified by the qualifier /DOLLARS, followed by a file specification of the file to be created. Insert the source program immediately after the command name. The source file is terminated either by another batch command or, if /DOLLARS has been specified, by the command $EOD.

2. At an interactive terminal, issue the PDS command CREATE followed by the file specification of the file to be created. Begin to input the source program at the beginning of the next line. Close the file by typing CTRL/Z.

The CREATE command is described in greater detail in Chapter 6, Section 6.4.1.1.

Examples:

1.
```
$CREATE/DOLLARS COBOL.CBL

.
.
.
00078      IF NF-DELIMITER = CR

00079         PERFORM READ-TRAN-LINE

00080         IF EOFFOUND GO TO G5999

00081         ELSE GO TO GS5

00082      IF CHAR-COUNT ZERO

00083         IF INMARKER < TRAN-LINE-LIMIT GO TO G25.

.

.

$EOD
```

2.
```
PDS> CREATE

FILE?  TEST.FTN

SUBROUTINE PROCI

C    FIRST DATA PROCESSING ROUTINE

C    COMMUNICATION REGION

COMMON/DTA/A(200),I

.

.

.

RETURN

END

CTRL/Z
```

8.4.2 The EDIT Command

The EDIT command allows the interactive user both to create and edit a source file via the Text Editor. Batch users should use the CREATE command to create a source file, which may be edited subsequently in either interactive or batch mode (See Chapter 7).

When the EDIT command specifies a non-existent file, the Line Text Editor creates one and prompts for input. For example:

```
PDS> EDIT

FILE?  NEWSOURCE.CBL

[CREATING NEW FILE]

INPUT
```

The user then begins to enter the source file beginning at the first position of the next line after 'INPUT'.

See Chapter 7 for details on how to use the Text Editor to edit the new file as it is being created.

To close the new file, the user must type carriage return as the first character in the line. This action causes the Editor to display an asterisk (*), which indicates that it expects an editor command rather than further input to the file because the command mode has changed from insert to edit. To close the file and exit to PDS, use the command EXIT. If the user wants to create further files, the EDIT command must be reissued.

Example:

```
PDS> EDIT

FILE?  TONY.FTN

[CREATING NEW FILE]

INPUT

      SUBROUTINE REPORT

C     INTERIM REPORT PROGRAM

C     COMMUNICATION REGION

      COMMON/DTA/A(200),

      RETURN

      END
<CR>

*EX

[EXIT]

PDS>
```

8.5 ERROR STATUS RETURNED TO PDS

When certain tasks exit it is possible for the system to notify PDS and hence the user of the 'worst' error found during execution. The system relies on the task using the Exit with Status Directive - see the Executive Reference Manual, Volume I. The status of the task is then recorded as one of

```
SUCCESS
WARNING
ERROR
SEVERE_ERROR
```

If exit with status is not implemented in the task, no status is recorded.

SUCCESS indicates that results should be as expected.

WARNING indicates that the task has succeeded but results may not be as expected.

ERROR is stronger than WARNING: results are unlikely to be as expected.

SEVERE_ERROR indicates one or more fatal errors or that the task was aborted.

If the task was invoked interactively, the termination message to the user's terminal includes the status.

If the task was invoked via an indirect command or in batch, the status can be used to control subsequent steps in the command file or batch job, see Section 8.5.1.

An Error status is returned by the PDS commands LINK, MACRO and LIBRARIAN.

Within PDS, failure to load a task, the operations MOUNT, DISMOUNT, ALLOCATE, DEALLOCATE, ASSIGN, DEASSIGN and failure to parse a command also return a status of SEVERE_ERROR to PDS.

8.5.1 Conditional Command Execution

Indirect Command files and Batch Command files can include the commands

```
ON error-severity action
GOTO label
STOP
CONTINUE
```

Of these, ON is a conditional command which relies on the status returned by certain tasks to PDS when they exit, or on PDS' own status return (see 8.5).

ON must be placed before the task(s) to which ON refers. For example, in a batch job,

```
$ON ERROR STOP
$MACRO MYPROG
$LINK MYPROG
$RUN MYPROG
```

Here, $ON has no effect on the MACRO assembly itself. If the assembly is completed with warnings only, the job continues with the Linking, and if the linking produces nothing worse then a warning, the task is run.

The action set by ON can be GOTO, STOP, CONTINUE, or any PDS command that would be valid in the command file. The setting remains in force until the next ON command, and is then superseded entirely. The initial (or default) setting is the example used above, [$]ON ERROR STOP. If an ON statement is found, on attempted execution, to be itself faulty, PDS reverts to the default setting.

NOTE

> ON cannot specify action to be taken during the execution of a task, nor actions dependent on the outcome of a previous task.

The command STOP prevents all further commands in the file or job being implemented.

CONTINUE is useful, for example, in overriding the default setting, thus

```
     ON SEVERE_ERROR CONTINUE
```

so that all later error statuses are ignored.

CONTINUE in an indirect or batch file does not imply previous suspension of a task, as it does in interactive or real-time use.

The label in a GOTO command must appear also, together with a colon, in front of a later command. For example,

```
     ...
     $ON WARNING GOTO ELSE
     $LINK MYPROG
     $RUN MYPROG
     $STOP
     $ELSE:  LINK OLDPROG
     $RUN OLDPROG
     $EOJ
```

Here MYPROG is linked and run, unless the link includes warning errors or worse, in which case OLDPROG is linked and run. If linking OLDPROG includes warning errors or worse, the setting of ON causes the command processor to look ahead for a label ELSE, and on finding none the job is abandoned.

STOP, CONTINUE and GOTO label can appear as separate commands, as well as in an ON statement.

CHAPTER 9

BASIC

9.1 INTRODUCTION

The information in this chapter relates to BASIC-11 only. Details of BASIC-PLUS-2 should be obtained from the BASIC-PLUS-2 specific documentation.

BASIC-11 provides immediate translation and storage of a user program while it is being input from an interactive terminal. The PDS user invokes the BASIC interpreter by typing the command BASIC. The BASIC system may not be used in batch mode under IAS.

The interactive nature of BASIC removes the need for separate steps in the development of a program. Once BASIC has been invoked, a program may be created, translated and run in a single session.

This chapter describes how to invoke BASIC, create and execute a program and then terminate a session. The following manuals describe the BASIC language itself:

> <u>BASIC-11 Language Reference Manual</u>
>
> <u>IAS BASIC User's Guide</u>

9.2 THE BASIC COMMAND

When the user issues the BASIC Command, BASIC displays as follows:

```
PDS> BASIC

IAS/RSX BASIC V02-01

READY
```

The text 'READY' indicates that BASIC is ready to receive a command or program line.

The BASIC command has no parameters or command qualifiers.

9.3 CTRL/C

If the user presses CTRL/C while a BASIC program is running, the system stops execution after the current line and dislays the number of the last line executed. The user may then issue further BASIC commands.

CTRL/C typed during the execution of a BASIC LIST or SAVE command or an immediate mode statement stops the execution of those commands or statements. It has no effect on the execution of other BASIC commands.

9.4 TERMINATING A BASIC SESSION

To terminate a BASIC session and return control to PDS, the user must type 'BYE' on a new line. The system then prints information about the session and prompts for further PDS commands. For example:

```
    BYE


    15.57.32 SIZE:14K CPU:10.24


    PDS>
```

9.5 EXAMPLE

```
    PDS> BASIC
    READY
    OLD MYBASIC
    LISTNH
    10 REM PROGRAM TO TRANSLATE MONTH NAMES TO NUMBERS
    50 T$ = "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
    100 PRINT "TYPE THE FIRST 3 LETTERS OF A MONTH";
    110 INPUT M$
    120 IF LEN (M$) <>3 GO TO 200
    130 M=(POS(T$,M$,1) + 2) /3
    140 REM CHECK IF MONTH IS SPELLED CORRECTLY
    150 IF M <> INT (M) GO TO 200
    160 PRINT M$" IS MONTH NUMBER"M
    170 GO TO 100
    200 PRINT "BAD MONTH" GO TO 100
    READY

    RUNNH
    TYPE THE FIRST 3 LETTERS OF A MONTH? NOV
    NOV IS MONTH NUMBER 11
    TYPE THE FIRST 3 LETTERS OF A MONTH? DEC
    DEC IS MONTH NUMBER 12
    TYPE THE FIRST 3 LETTERS OF A MONTH? JAN
    JAN IS MONTH NUMBER 1
    TYPE THE FIRST 3 LETTERS OF A MONTH? AUD
    BAD MONTH
    TYPE THE FIRST 3 LETTERS OF A MONTH?  CTRL/C
    STOP AT LINE 110
    READY
    BYE
    12.39.27 SIZE:14K CPU:0.76
    PDS>
```

In this example the user first invokes BASIC by issuing the BASIC command. BASIC indicates that it is ready to accept BASIC program lines and commands by printing READY. The user then retrieves an existing BASIC program by entering the OLD command. This program is printed and executed by the LIST and RUN commands respectively. Since this program is written as a loop, that is, after executing line 200 it loops back to line 100, it will execute indefinitely. By entering CTRL/C the user terminates the program execution. BASIC then prints the number of the line at which execution was stopped. The BYE command terminates the BASIC session.

CHAPTER 10

COBOL

A COBOL programmer must complete four steps to transform a COBOL source program into an executing task:

1. Create one or more source files;
2. Compile the source files;
3. Link the compiled, i.e. object, files; and
4. Run the executable task.

This chapter describes how to use IAS commands to perform these steps. See Chapter 5 for a description of the SUBMIT command, which allows the user to create a file of IAS commands to be submitted to a batch stream. Consult the following manuals for information about programming in COBOL on PDP-11 machines:

PDP-11 COBOL Language Reference Manual

PDP-11 COBOL User's Guide

10.1 CREATING SOURCE FILES

Either the CREATE command or the EDIT command may be used to create source files. See Chapter 8, Section 8.3. The EDIT command has the advantage that it allows the interactive user immediate access to all its editing facilities. To correct errors made while using the CREATE command, however, the user must rely on keyboard facilities or close the file and then issue the EDIT command.

10.2 THE COBOL COMMAND

By default, the COBOL command compiles a source program, and produces an object file. For example:

```
PDS> COBOL

FILE? SOURCE.CBL
```

This command string compiles the program SOURCE.CBL. and produces an object file named SOURCE.OBJ. If the user omits the file type field in the specification of the source file, the COBOL complier assumes it to be CBL.

10.2.1 Compiling COBOL Source Files

Only one source file may be specified with each COBOL command. The following command strings all compile the source file COBSRC.CBL.

1. PDS> COBOL

   FILE? COBSRC

2. $COBOL COBSRC

3. PDS> COBOL COBSRC

Each of the command strings above instructs the system to compile the source file specified and to produce compiler output as the defaults dictate.

By default, the compiler:

1. Produces an object file which is given the name of the source file and OBJ as the filetype.

2. Compiles the source file according to the compiler's default switches. (See the COBOL command specification in Part 2 for a description of the compiler switches.)

3. If file processing features which use RMS-11 facilities are used, the compiler produces a skeleton .ODL file that should be used when linking non-trivial COBOL programs. This facility is described in detail in the COBOL User's Guide.

10.2.2 COBOL Command Qualifiers

The qualifiers to the COBOL command are:

```
/OBJECT[:filespec]
/NOOBJECT
/LIST[:filespec]
/NOLIST
/SWITCHES:(switches)
```

The compiler produces an object file unless the user specifies /NOOBJECT. The object file may be named by default or given a name specified after /OBJECT. See the command specification in Part 2 for further details.

Users must specify /LIST to obtain a listing. The /LIST:filespec qualifier allows the user to send the listing to a file; otherwise, the listing file is printed at the line printer and then deleted.

The qualifier /SWITCHES is described in the next section.

10.2.3 Compiler Switches

The PDP-11 COBOL compiler provides switches to tailor compilation to particular needs. The user specifies the switches by means of the /SWITCHES qualifier to the COBOL command. For example:

```
$COB/SWITCHES:(/MAP)     SOURCE.CBL
```

The switch /MAP tells the compiler to produce a map listing.

The specified switches must be enclosed in parentheses. For example:

```
PDS> COBOL/SWITCHES(/ERR:2/MAP/CVF)/LIST:ACCOUNT.LST

FILE? ACCTS.CBL
```

When the user does not specify any switches, the compiler operates according to defaults. The default switches are:

```
(/ERR:0/ACC:1/NOMAP)
```

The COBOL command specification in Part 2 defines all the possible compiler switches.

10.2.4 Compiler Error Messages

The compiler generates error messages (diagnostic, warning and fatal) whenever it detects an error in the source program. With some exceptions, a source error detected by the compiler results in the associated message being embedded within the source program listing. That is, when an error is detected in the source program, the compiler prints the error message either before or after the erroneous source program statement.

See the PDP-11 COBOL User's Guide for a detailed description of error messages.

10.3 LINKING OBJECT FILES

The user issues the LINK command to link COBOL object files to create an executable task.

10.3.1 The LINK Command

The LINK command invokes the IAS Task Builder to build an executable task from object files generated by the COBOL command and from object modules held in user-written and system library files (see Chapter 8, Section 8.2). In particular the system object module libraries COBLIB.OLB and RMSLIB.OLB must be specified.

The IAS Task Builder Reference Manual contains a complete description of the Task Builder.

This section gives information to help the programmer use the LINK command. The user modifies the action of the Task Builder by specifying or defaulting various options.

To link one or more COBOL programs using the system default Task Builder switches and options, the user issues the LINK command followed by the list of object files to be linked together into an executable task.

For examples:

```
     LINK  PRODUCTS  STOCKS [1,1]COBLIB/LI [1,1]RMSLIB/LI
```

links together the COBOL object files PRODUCTS.OBJ and STOCKS.OBJ.

10.3.1.1 Options - The qualifier /OPTIONS allows the user to specify Task Builder options. In interactive mode the presence of the qualifier /OPTIONS in the command qualifier list causes the Task Builder to prompt OPTIONS? after the input files have been specified. For example:

```
     PDS> LINK/OPTIONS

     FILE?  PROG.OBJ,REPORT.OBJ,[1,1]COBLIB/LI,[1,1]RMSLIB/LI

     OPTIONS?
```

The user then enters the options one line at a time. A slash (/) as the first character in a line then terminates the option input and the Task Builder resumes execution.

For example:

```
     PDS> LINK/OPTIONS

     FILE?  MYCOB.OBJ, PROG.OBJ, [1,1]COBLIB/LI, [1,1]RMSLIB/LIB

     OPTIONS?  UNITS=9

     OPTIONS?  ASG=DT1:7:8:9

     OPTIONS?  /
```

In batch mode, the presence of the /OPTIONS qualifier in the command qualifier list causes the Task Builder to expect one or more options to be specified on one or more lines immediately following the command string. A line containing a slash (/) in the first character position terminates the list of options.

For example:

```
     $LINK/OPTIONS PROG REPORT [1,1]COBLIB/L [1,1]RMSLIB/L

     UNITS=9

     ASG=DT1:7:8:9

     /
```

The Task Builder options are summarized in a table in the LINK command in Part 2.

10.3.1.2 Object Module Libraries - The file qualifier /LIBRARY specifies a library file that contains the user-written object modules to be incorporated in the task. The Task Builder automatically searches system object module libraries for referenced modules.

In addition, the supplied object module libraries COBLIB.OLB and RMSLIB.OLB must always be specified when listing COBOL programs and must be specified in that order.

Example:

```
     PDS> LINK CBLPROG [1,1]COBLIB/LI [1,1]RMSLIB/LI
```

If the .ODL file generated by the COBOL compiler or a user supplied .ODL file is used for complex structured programs then the library specifications for COBLIB and RMSLIB must be included in the .ODL file. See the COBOL User's Guide for further details.

10.3.1.3 Output Files - The Task Builder does not generate any output files, other than an executable task image, unless the user specifically requests them by supplying the relevant qualifiers. The possible output files and the associated qualifiers are:

Output File	Qualifier
Task image file	/TASK[:filespec]
Memory allocation map file	/MAP:filespec]
Symbol definition file	/SYMBOLS[:filespec]

10.3.1.4 Example - The following example links three object files.

```
     PDS> LINK/TASK:WAGES/MAP:WAGES/OPTIONS

     FILES? PAY, PEOPLE, MONTH, [1,1]COBLIB/LI, [1,1]RMSLIB/LI

     OPTIONS? UNITS = 5

     OPTIONS? ASG=DT2:1:2,TI:3,MT:4:5

     OPTIONS? /

     PDS>
```

The LINK command links the three object files to create a task image file named WAGES.TSK and a map file named WAGES.MAP.

10.4 RUNNING THE TASK

A COBOL programmer compiles and links a task in separate operations. The RUN command is then used to execute the task image created by the LINK command.

To run a linked COBOL task, issue the RUN command and specify the task image file generated by the LINK command.

Examples:

```
     1.   PDS> RUN
          FILE? WAGES

     2.   $RUN WAGES
```

Both examples instruct the system to run the task named CALC.TSK.

CHAPTER 11

FORTRAN

A FORTRAN programmer must complete four steps to transform a FORTRAN source program into an executing task:

1. Create one or more source files;

2. Compile the source files;

3. Link the compiled source files, i.e. object files; and

4. Run the executable task.

This chapter describes how to use IAS commands to perform these steps.

See Chapter 5 for a description of the SUBMIT command, which allows the user to submit a file of IAS commands to a batch stream. A user could create such a file to compile, link and run his task in a single batch job.

Consult the following manuals for information about programming in FORTRAN IV or FORTRAN IV-PLUS:

IAS/RSX-11 FORTRAN IV User's Guide
FORTRAN IV-PLUS User's Guide
PDP-11 FORTRAN Language Reference Manual

11.1 CREATING SOURCE FILES

Either the CREATE command or the EDIT command may be used to create source files. See Chapter 8, Section 8.3. The EDIT command has the advantage that it allows the user, immediate access to all its editing facilities. To correct errors made while using the CREATE command, however, the user must rely on keyboard facilities or close the file and then issue the EDIT command.

11.2 THE FORTRAN COMMAND

The basic function of the FORTRAN command is to compile one or more FORTRAN source programs. Command qualifiers, including compiler switches and options, determine the form of the output to be generated by the compiler.

11.2.1 Compiling Source Files

Only one source file may be specified with each FORTRAN command. The following command strings all compile the source file INVERT.FTN.

1.
   ```
   PDS> FORTRAN

   FILE? INVERT
   ```

2.
   ```
   $FORTRAN INVERT
   ```

3.
   ```
   PDS> FORTRAN INVERT
   ```

Each of the command strings above instructs the system to compile the source file specified and to produce compiler output as the defaults dictate.

By default, the compiler:

1. Produces an object file which is given the name of the source file and the OBJ as the filetype.

2. Compiles the source file according to the compiler's default switches. (See the FORTRAN command specification in Part 2 for a description of the compiler switches.)

11.2.2 FORTRAN Command Qualifiers

Command qualifiers, each preceded by a slash (/), immediately follow the command name. For example:

```
PDS> FORTRAN/LIST/OBJECT/SWITCHES:(/CK) SOURCE.FTN
```

A programmer specifies command qualifiers in order to modify the function of the FORTRAN command according to the needs of the program. Qualifiers may also be specified merely to affirm default compiler actions. For instance, the example above specifies /OBJECT even though the FORTRAN command produces an object file by default. (See Section 11.2.1 for a list of compiler defaults.)

Compiler switches are listed after the /SWITCHES: qualifier. The list of switches must be enclosed in parentheses. The slash preceding each switch separates one from the next within the list. For example:

```
$FORTRAN/SWITCHES:(/CK/CO=7/TR:LINES) PROG1.FTN
```

The possible switches depend on whether the programmer is using FORTRAN IV or FORTRAN IV-PLUS. Both sets of switches are listed in the specification of the FORTRAN command in Part 2.

11.2.3 Examples

The following commands all compile a FORTRAN source file:

1. `$FORTRAN/OBJECT/LIST:PRINT    RDIN`

   Compile the source program RDIN.FTN, create an object file name RDIN.OBJ and create a listing file called PRINT.LST.

2. `$FORTRAN/OBJECT/LIST:LPROC1    PROC1`

   Compile the source program PROC1.FTN, create an object file named PROC1.OBJ and create a listing file called LPROC1.LST.

3. `$FORTRAN RPRT.FTN`

   Compile the source program RPRT.FTN to create an object file named RPRT.OBJ.

Note that the file specifications to the /LIST qualifier need not include a filetype. In this case, the system assumes the filetype to be LST.

11.3 LINKING OBJECT FILES

The user issues the LINK command to link FORTRAN object files to create an executable task.

11.3.1 The LINK Command

The LINK command invokes the IAS Task Builder to build an executable task from object files generated by the FORTRAN command and from object modules held in user-written and system library files (see Chapter 8, Section 8.2).

The IAS Task Builder Reference Manual contains a complete description of the Task Builder.

This section gives information to help the programmer use the LINK command. The user modifies the action of the Task Builder by specifying or defaulting various options.

To link one or more FORTRAN programs using the system default Task Builder switches and options, the user issues the LINK command followed by the list of object files to be linked together into an executable task.

For examples:

```
     LINK     PERFECT     NUMBER
```

links together the FORTRAN object files PERFECT.OBJ and NUMBER.OBJ.

11.3.1.1 Options - The qualifier /OPTIONS allows the user to specify Task Builder options. In interactive mode the presence of the qualifier /OPTIONS in the command qualifier list causes the Task Builder to prompt OPTIONS? after the input files have been specified.

For example:

```
PDS> LINK/OPTIONS

FILE?  PROG.OBJ,REPORT.OBJ

OPTIONS?
```

The user then enters the options one line at a time. A slash (/) as the first character in a line then terminates the option input and the Task Builder resumes execution.

For example:

```
PDS> LINK/OPTIONS

FILE?  FORT.OBJ, PROG.OBJ

OPTIONS?  ACTFIL=8

OPTIONS?  MAXBUF=160

OPTIONS?  UNITS=9

OPTIONS?  ASG=DT1:7:8:9

OPTIONS?  /
```

In batch mode, the presence of the /OPTIONS qualifier in the command qualifier list causes the Task Builder to expect one or more options to be specified on one or more lines immediately following the command string. A line containing a slash (/) in the first character position terminates the list of options.

For example:

```
$LINK/OPTIONS   PROG.OBJ,    REPORT.OBJ

ACTFIL=8

MAXBUF=160

UNITS=9

ASG=DT1:7:8:9

/
```

The Task Builder options are summarized in a table in the LINK command in Part 2. The table indicates with an 'F' the options that are relevant to FORTRAN programs.

11.3.1.2 Object Modules - The file qualifier /LIBRARY specifies a library file that contains the user-written object modules to be incorporated in the task. The Task Builder automatically searches system object module libraries for referenced modules.

Example:

```
$LINK (FORT.LIB/LIBRARY:(MOD1,MOD2), FORTRAN.OBJ)
```

11.3.1.3 Output Files - The Task Builder does not generate any output files, other than an executable task image, unless the user specifically requests them by supplying the relevant qualifiers. The possible output files and the associated qualifiers are:

Output File	Qualifier
Task image file	/TASK[:filespec]
Memory allocation map file	/MAP:filespec]
Symbol definition file	/SYMBOLS[:filespec]

11.3.1.4 Example - The following example links three object files.

```
PDS> LINK/TASK:CALC/MAP:CALC/OPTIONS

FILES? RDIN.OBJ,  PROC1.OBJ,  RPRT.OBJ

OPTIONS? UNITS = 5

OPTIONS? ASG=DT2:1:2,TI:3,MT:4:5

OPTIONS? /

PDS>
```

The LINK command links the three object files to create a task image file named CALC.TSK and a map file named CALC.MAP.

11.4 RUNNING THE TASK

A FORTRAN programmer compiles and links a task in separate operations. The RUN command is then used to execute the task image created by the LINK command.

To run a linked FORTRAN task, issue the RUN command and specify the task image file generated by the LINK command.

Examples:

1.
```
PDS> RUN
FILE? CALC
```

2.
```
$RUN CALC
```

Both examples instruct the system to run the task named CALC.TSK.

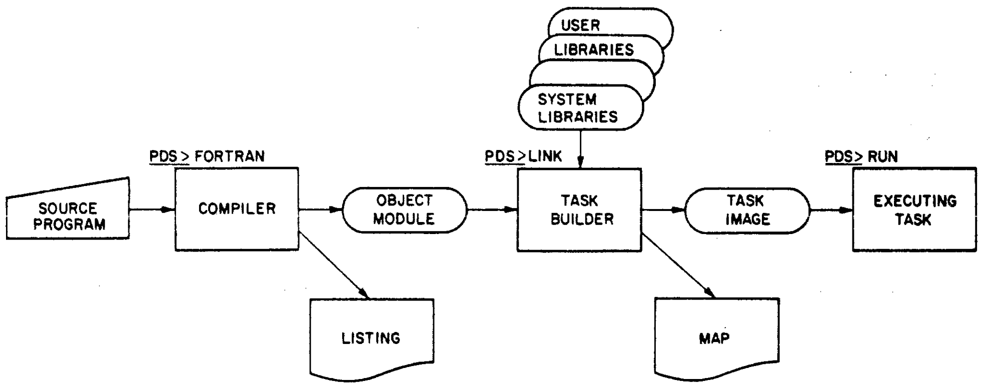


Figure 11-1
Steps in Creating a FORTRAN Program

CHAPTER 12

MACRO

A MACRO-11 programmer must complete four steps to transform a MACRO-11 program into an executing task:

1. Create one or more source files,
2. Assemble the source files,
3. Link the assembled, i.e. object, files, and
4. Run the executable task.

This chapter describes how to use IAS commands to perform these steps. It also introduces the On-line Debugging Technique (ODT), a system program which aids in debugging assembled and linked object programs (Section 12.5). Consult the IAS/RSX-11 MACRO-11 Reference Manual for information about programming in MACRO-11.

See Chapter 5 for a description of the SUBMIT command. It allows the user to submit a file of IAS commands to a batch processor. A user could create such a file to compile, link and run his task in a single batch job.

12.1 CREATING SOURCE FILES

Either the CREATE command or the EDIT command may be used to create source files. See Chapter 8, Section 8.3. The EDIT command has the advantage that it allows the user immediate access to all its editing facilities. To correct errors made while using the CREATE command, however, the user must rely on keyboard facilities or close the file and then issue the EDIT command.

12.2 THE MACRO COMMAND

The MACRO command assembles one or more ASCII source files containing MACRO-11 statements into a single relocatable binary object file. Command qualifiers, including assembler switches, determine the output to be generated by the assembler.

12.2.1 Assembling Source Files

The following command string assembles the source files LOCATE.MAC and FIND.MAC:

1. PDS> MAC

   FILES? LOCATE+FIND

2. $MACRO LOCATE+FIND

Each of the command strings above instructs the system to assemble the source files specified and to produce assembler output as the defaults dictate. Note that the MACRO command requires the source files to be concatenated with a plus sign (+); the assembler does not accept the more common list format, that is, a list enclosed in parentheses, with list items separated by a comma, spaces or tabs. By default, the assembler produces an object file which is given the name of the last source file specified but with OBJ as the filetype.

12.2.2 Command and File Qualifiers

Command qualifiers, each preceded by a slash (/), immediately follow the command name.

For example:

```
PDS> MACRO/LIST/OBJECT    LOCATE+FIND
```

The programmer specifies file qualifiers immediately after the relevant file specification. For example:

```
$MAC MACLIB.MLB/LIB+TEST
```

The LIBRARY qualifier instructs the assembler to treat MACLIB.MLB as a macro library file. The /LIST qualifier requests a listing to be sent to the line printer.

A programmer specifies command and file qualifiers in order to modify the function of the MACRO command according to the needs of the program. Qualifiers may also be specified merely to affirm default assembler actions. For instance, the first example above specifies /LIST and /OBJECT even though the MACRO command produces an object file by default. (See Section 12.2.1 for a list of assembler defaults.)

The specification of the MACRO command in Part 2 lists all the possible command and file qualifiers. Programmers should consult the IAS/RSX-11 MACRO-11 Reference Manual for a full description.

Example:

```
PDS> MACRO/OBJECT:FINAL

FILE? ROUT.MAC+MAIN.MAC
```

Assemble the source programs ROUT.MAC and MAIN.MAC to produce an object file named FINAL.OBJ.

12.3 LINKING OBJECT FILES

The user issues the LINK command to link MACRO-11 object files to create an executable task. See Section 12.5 for information about debugging linked object programs.

12.3.1 The LINK Command

The LINK command invokes the IAS Task Builder to build an executable task from object files generated by the FORTRAN or MACRO command and from object modules held in user-written and system library files (see Section 12.3.1.3).

The IAS Task Builder Reference Manual contains a complete description of the Task Builder. This section gives information to help the programmer use the LINK command.

The user may modify the action of the Task Builder by specifying various options. To link one or more MACRO-11 programs with the system default Task Builder switches and options, the user issues the LINK command followed by the list of object files to be linked together into an executable task.

For example:

```
$LINK     REALTIME     ADCONVERT
```

links together the object programs REALTIME.OBJ and ADCONVERT.OBJ.

12.3.1.1 Options - The /OPTIONS qualifier allows the user to specify Task Builder options. In interactive mode the presence of the qualifier /OPTIONS in the command qualifier list causes the Task Builder to prompt OPTIONS? after the input files have been specified. For example:

```
PDS> LINK/OPTIONS

FILE?  PROG.OBJ, REPORT.OBJ

OPTIONS?
```

The user then enters the options one line at a time. A slash (/) as the first character in a line then terminates the list of options and the Task Builder resumes executing.

For example:

```
PDS> LINK/OPTIONS

FILE? MAIN.OBJ,   PROG.OBJ

OPTIONS? TASK=SYSMAN

OPTIONS? UIC=[1,1]

OPTIONS? LIBR=SYSRES:RO

OPTIONS? /
```

In batch mode, the presence of the /OPTIONS qualifier in the command qualifier list causes the Task Builder to expect one or more options to be specified on one or more lines immediately following the command string. The user must specify a single option on each line. A card or line containing a slash (/) in the first character position terminates the list of options.

For example:

```
$LINK/OPTIONS   PROG.OBJ,   REPORT.OBJ

TASK=SYSMAN

UIC=[1,1]

LIBR=SYSRES:RO

/
```

The Task Builder options are summarized in the specification of the LINK command in Part 2. The summary marks the options that are relevant to MACRO programs with the letter M.

12.3.1.2 Object Modules - The file qualifier /LIBRARY specifies the library files that contain the user-written object modules to be incorporated in the task. The Task Builder automatically searches system object module libraries for referenced modules.

Example:

```
$LINK  MACRO.LIB/LIBRARY:(MAC1, MAC2) MACRO.OBJ
```

12.3.1.3 Output Files - The Task Builder does not generate any output files, other than an executable task image, unless the user specifically requests them by supplying the relevant qualifiers. The possible output files and the associated qualifiers are:

Output File	Qualifier
Task image file	/TASK[:filespec]
Memory allocation map file	/MAP[:filespec]
Symbol definition file	/SYMBOLS[:filespec]

12.3.1.4 Example - The following example links three object files to form a task named CALC.TSK.

```
PDS> LINK/TASK:CALC/MAP:CALC/DEBUG/OPTIONS

FILE? (SEG1.OBJ, SEG2.OBJ, MACRO.OBJ)

OPTIONS? UNITS = 5

OPTIONS? ASG=DT2:1:2,TI:3,MT:4:5

OPTIONS? /

PDS>
```

The command string above links the three object files to create a task image file named CALC.TSK and a map file named CALC.MAP. The /DEBUG qualifier instructs the Task Builder to include a debugging aid (i.e. the ODT program, see Section 12.5.1) and Task Builder options assign logical unit numbers.

12.4 RUNNING THE TASK

A MACRO-11 programmer assembles and links a task in separate operations. The RUN command is then used to begin execution of the task image created by the LINK command.

When used to execute a MACRO-11 task, the RUN command has no qualifiers and only one parameter, the file specification of the task to be run. The file containing the executable task is the task image file generated by LINK.

Examples:

1.
   ```
   PDS> RUN

   FILE? CALC.TSK
   ```

2.
   ```
   $RUN CALC.TSK
   ```

Both examples instruct the system to run the task named CALC.TSK.

12.5 DEBUGGING

12.5.1 The On-line Debugging Technique

IAS provides the On-line Debugging Technique (ODT) to help programmers debug linked and assembled object programs. To incorporate ODT in the linked program, the programmer specifies the /DEBUG qualifier to the LINK command (see Section 12.3.1.1).

For example:

```
$LINK/DEBUG    MACRO.OBJ
```

The Task Builder then automatically includes ODT in the task image.

The IAS/RSX-11 ODT Reference Manual contains a complete description of ODT. In brief, however, the programmer interacts with ODT and the object program from an interactive terminal to:

1. Print the contents of any location for examination or alteration.

2. Run all or any portion of the object program using the breakpoint feature.

3. Search the object program for specific bit patterns.

4. Search the object program for words which reference a specific word.

5. Calculate a block of words or bytes with a designated value.

6. Fill a block of words or bytes with a designated value.

The breakpoint is one of ODT'S most useful features. When debugging a program, it is often desirable to allow the program to run normally up to a predetermined point, at which time the contents of various registers or locations can be examined and possibly modified. To accomplish this, ODT acts as a monitor to the user program.

During a debugging session you should have the current assembly listing and memory allocation map of the program to be debugged with you at the terminal. Minor corrections to the program may be made on-line during the debugging session. The program may then be run under control of ODT to verify any changes made. Major corrections, however, such as a missing subroutine, should be noted on the assembly listing and incorporated in a subsequent updated program assembly.

12.5.2 User-Written Debugging Aids

A programmer may also incorporate a user-written debugging aid in a linked object program. The file containing the debugging aid is specified with the /DEBUG qualifier.

For example:

```
PDS> LINK/DEBUG:[1,1]DDT/READ_WRITE/SYMBOLS

FILES?  MACRO.OBJ
```

CHAPTER 13

CORAL

A CORAL programmer must complete four steps to transform a CORAL source program into an executing task:

1. Create one or more source files;
2. Compile the source files;
3. Link the compiled, i.e. object, files; and
4. Run the executable task.

This chapter describes how to use IAS commands to perform these steps.

See Chapter 5 for a description of the SUBMIT command, which allows the user to submit a file of IAS commands to a batch stream. A user could create such a file to compile, link and run his task in a single batch job.

Consult the following manual for information about programming in CORAL:

> PDP-11 CORAL 66 (with FPP support) Language Reference Manual and User's Guide.

13.1 CREATING SOURCE FILES

Either the CREATE command or the EDIT command may be used to create source files. See Chapter 8, Section 8.3. The EDIT command has the advantage that it allows the user, immediate access to all its editing facilities. To correct errors made while using the CREATE command, however, the user must rely on keyboard facilities or close the file and then issue the EDIT command.

13.2 THE CORAL COMMAND

The basic function of the CORAL command is to compile one or more CORAL source programs. Command qualifiers, including compiler switches and options, determine the form of the output to be generated by the compiler.

13.2.1 Compiling Source Files

Only one source file may be specified with each CORAL command. The following command strings all compile the source file INVERT.COR.

1. PDS> CORAL

   FILE? INVERT

2. $CORAL INVERT

3. PDS> CORAL INVERT

Each of the command strings above instructs the system to compile the source file specified and to produce compiler output as the defaults dictate.

By default, the compiler:

1. Produces an object file which is given the name of the source file and OBJ as the filetype.

2. Compiles the source file according to the compiler's default switches. (See the CORAL command specification in Part 2 for a description of the compiler switches.)

13.2.2 CORAL Command Qualifiers

Command qualifiers, each preceded by a slash (/), immediately follow the command name. For example:

```
PDS> CORAL/LIST/OBJECT/SWITCHES:(/BC) SOURCE.FTN
```

A programmer specifies command qualifiers in order to modify the function of the CORAL command according to the needs of the program. Qualifiers may also be specified merely to affirm default compiler actions. For instance, the example above specifies /OBJECT even though the CORAL command produces an object file by default. (See Section 13.2.1 for a list of compiler command defaults.)

Compiler switches are entered after the /SWITCHES: qualifier. The list of switches must be enclosed in parentheses. The slash preceding each switch separates one from the next within the list. For example:

```
$CORAL/SWITCHES:(/BC/OP:2/LP:3) PROG1.COR
```

The switches are listed in the specification of the CORAL command in Part 2.

13.2.3 Examples

The following commands all compile a CORAl source file:

1. `$CORAL/OBJECT/LIST:PRINT    RDIN`

   Compile the source program RDIN.COR, create an object file name RDIN.OBJ and create a listing file called PRINT.LST.

2. `$CORAL/OBJECT/LIST:LPROC1    PROC1`

   Compile the source program PROC1.COR, create an object file named PROC1.OBJ and create a listing file called LPROC1.LST.

3. `$CORAL/OBJECT/LIST:READ RPRT.COR`

   Compile the source program RPRT.COR, create an object file named RPRT.OBJ and create a listing file called READ.LST.

Note that the file specifications to the /LIST qualifier need not include a filetype. In this case, the system assumes the filetype to be LST.

13.3 LINKING OBJECT FILES

The user issues the LINK command to link CORAL object files to create an executable task.

13.3.1 The LINK Command

The LINK command invokes the IAS Task Builder to build an executable task from object files generated by the CORAL command and from object modules held in user-written and system library files (see Chapter 8, Section 8.2).

The IAS Task Builder Reference Manual contains a complete description of the Task Builder.

This section gives information to help the programmer use the LINK command. The user modifies the action of the Task Builder by specifying or defaulting various options.

To link one or more CORAL programs using the system default Task Builder switches and options, the user issues the LINK command followed by the list of object files to be linked together into an executable task.

For examples:

```
LINK     PERFECT     NUMBER
```

links together the CORAL object files PERFECT.OBJ and NUMBER.OBJ.

13.3.1.1 Options - The qualifier /OPTIONS allows the user to specify Task Builder options. In interactive mode the presence of the qualifier /OPTIONS in the command qualifier list causes the Task Builder to prompt OPTIONS? after the input files have been specified.

For example:

```
PDS> LINK/OPTIONS

FILE?  PROG.OBJ,REPORT.OBJ

OPTIONS?
```

The user then enters the options one line at a time. A slash (/) as the first character in a line then terminates the option input and the Task Builder resumes execution.

For example:

```
PDS> LINK/OPTIONS

FILE?  CORAL.OBJ, PROG.OBJ

OPTIONS?  ACTFIL=8

OPTIONS?  MAXBUF=280

OPTIONS?  UNITS=9

OPTIONS?  ASG=DT1:7:8:9

OPTIONS?  /
```

In batch mode, the presence of the /OPTIONS qualifier in the command qualifier list causes the Task Builder to expect one or more options to be specified on one or more lines immediately following the command string. A line containing a slash (/) in the first character position terminates the list of options.

For example:

```
$LINK/OPTIONS   PROG.OBJ,    REPORT.OBJ

ACTFIL=8

MAXBUF=280

UNITS=9

ASG=DT1:7:8:9

/
```

The Task Builder options are summarized in a table in the LINK command in Part 2. The table indicates with a 'C' those that are relevant to CORAL programs. Note that the MAXBUF and FMTBUF options have a special meaning when linking CORAL programs. For a full description see the PDP-11 CORAL 66 manual, Chapter 7.

13.3.1.2 Object Modules - The file qualifier /LIBRARY specifies a library file that contains the user-written object modules to be incorporated in the task. The Task Builder automatically searches system object module libraries for referenced modules.

Example:

```
$LINK (COROTS.OLB/LIBRARY:(MOD1,MOD2), CORAL.OBJ)
```

13.3.1.3 Output Files - The Task Builder does not generate any output files, other than an executable task image, unless the user specifically requests them by supplying the relevant qualifiers. The possible output files and the associated qualifiers are:

Output File	Qualifier
Task image file	/TASK[:filespec]
Memory allocation map file	/MAP:[filespec]
	or
	/MAP:(filespec/qualifier)
Symbol definition file	/SYMBOLS[:filespec]

The MAP filespec qualifiers can be /FILES, /FULL, /NARROW, /SHORT, /WIDE.

13.3.1.4 Example - The following example links three object files.

```
PDS> LINK/TASK:CALC/MAP:CALC/OPTIONS

FILES? RDIN.OBJ,  PROC1.OBJ,  RPRT.OBJ

OPTIONS? UNITS = 5

OPTIONS? ASG=DT2:1:2,TI:3,MT:4:5

OPTIONS? /

PDS>
```

The LINK command links the three object files to create a task image file named CALC.TSK and a map file named CALC.MAP.

13.4 RUNNING THE TASK

A CORAL programmer compiles and links a task in separate operations. The RUN command is then used to execute the task image created by the LINK command.

To run a linked CORAL task, issue the RUN command and specify the task image file generated by the LINK command.

Examples:

1.
   ```
   PDS> RUN
   FILE? CALC
   ```

2.
   ```
   $RUN CALC
   ```

Both examples instruct the system to run the task named CALC.TSK.

APPENDIX A

OBJECT MODULE PATCH UTILITY (PAT)

A.1 INTRODUCTION TO PAT

The object module patch utility PAT, allows you to patch, or update, code in a relocatable binary object module.

PAT accepts a file containing corrections or additional instructions and applies these to the original object module.

Correction input is prepared in source form and assembled by the MACRO-11 assembler. PAT allows you to increase the size of object modules because changes are applied before the module is linked by the Task Builder.

Input to PAT is two files, the original input file and a correction file containing the corrections and additions to the input file. The input file consists of one or more concatenated object modules. You may correct only one of these object modules with a single execution of the PAT utility. The correction file consists of object code that, when linked by the Task Builder, either overlays or is added to the original object module.

Output from PAT is the updated input file.

Figure A-1 shows how PAT updates a file (FILE1) consisting of three object modules (MOD1, MOD2, and MOD3) by appending a correction file to MOD2. The updated module is then relinked with the rest of the file by the Task Builder to produce an executable task.

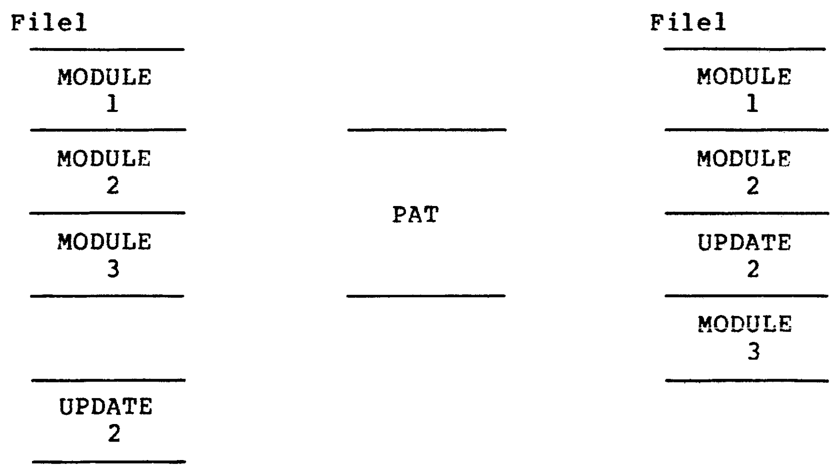


Figure A-1 Updating a Module Using PAT

There are several steps involved in using PAT to update a file. First, create the correction file using a text editor. The correction file must then be assembled to produce an object module. The input file and the correction file in object module form are then submitted to PAT for processing. Finally, the updated object module, along with the object modules that comprise the rest of the file, can be submitted to the Task Builder to resolve global symbols and create an executable task. Figure A-2 shows the processing steps involved in generating an updated task file using PAT.

1. Generate a correction file using the Text Editor.

2. Execute the assembler (or compiler) to generate an object module version of the file.

3. Execute PAT using as input the correction file and the module to be updated.

4. Execute the Task Builder to resolve new addresses and generate an executable task.

Figure A-2 Processing Steps Required to Update a Module Using PAT

A.2 SPECIFYING THE PAT COMMAND STRING

PAT is installed in the IAS system as a system library task ($$$PAT). As such, you can invoke PAT by entering the three character command "PAT". The command line indicating what actions PAT is to perform is entered either on the same line:

```
     PDS> PAT command line
```

or by typing carriage return and allowing the utility to prompt for its command line.

```
     PDS> PAT <carriage return>

     PAT> command line

     PAT> ^Z
```

Specify the PAT command line in the following form:

```
     [outfile]=infile[/CS[:number]],correctfile[/CS[:number]]
```

where:

outfile	is the file specification for the output file. If you don't specify an output file, none is generated.
infile	is the file specification for the input file. This file can contain one or more concatenated object modules.
correctfile	is the file specification for the correction file. This file contains the updates being applied to a single module in the input file.
CS	specifies the Checksum switch, which directs PAT to generate an octal value for the sum of all the binary data comprising the module in the file to which the switch is applied. (Refer to the section "Determining and Validating the Contents of Files," for information on how to use /CS.)
number	specifies an octal value that directs PAT to compare the checksum value it computes for a module with the octal value you specify as number.

A.3 HOW PAT APPLIES UPDATES

PAT applies updates to a base input module using additions and corrections supplied in a correction file. This section describes the PAT input and correction files, gives information on how to create the correction file, and gives examples of how to use PAT.

A.3.1 The Input File

The input file is the file to be updated; it is the base for the output file. The input file must be in object module format. When PAT executes, the correction file module is applied to this file.

A.3.2 The Correction File

The correction file must also be in object module format. It is usually created from a MACRO-11 source file in the following format:

```
.TITLE inputname
.IDENT updatenum
inputline
inputline
    *

    *

    *
```

where:

inputname	is the name of the module to be corrected by the PAT update. That is, inputname must be the same name as the name specified on the input file .TITLE directive for a single module in the input file.
updatenum	is any value acceptable to the MACRO-11 assembler. Generally, this value reflects the update version of the file being processed by PAT, as shown in the examples below.
inputline	are lines of input to be used to correct and update the input file.

During PAT execution, new global symbols defined in the correction file are added to the module's symbol table. Duplicate global symbols in the correction file supersede their counterparts in the input file, provided both definitions are relocatable or absolute.

A duplicate PSECT or CSECT supersedes the previous PSECT or CSECT, provided:

- Both have the same relocatability attribute (ABS or REL).
- Both are defined with the same directive (.PSECT or .CSECT).

If PAT encounters duplicate PSECT names, the length attribute for the PSECT is set to the length of the longer PSECT and a new PSECT is appended to the module.

If a transfer address is specified, it supersedes that of the module being patched.

A.3.3 Creating the Correction File

The first step is to generate the correction file. Use any editor to generate source additions and corrections to your file.

The correction file must be translated into object module format before it can be processed by PAT.

A.3.4 How PAT and the Task Builder Update Object Modules

The following examples show an input file and a correction file to be processed by PAT and Task Builder, along with a source-like representation of what the output file would look like once PAT and Task Builder complete processing. Two techniques are described, one for overlaying lines in a module and the other for appending a subroutine to a module.

Overlaying Lines in a Module

The first example illustrates a technique for overlaying lines in a module using a patch file. First, PAT appends the correction file to the input file. The Task Builder is then executed to replace code within the input file. The input file for this example is:

```
        .TITLE  ABC
        .IDENT  /01/
ABC::
        MOV     A,C                     ;
        CALL    XYZ                     ;
        RETURN                          ;
         .END
```

To add the instruction ADD A,B after the CALL instruction, include the following patch source file:

```
        .TITLE  ABC
        .IDENT  /01.01/
.=.+12
        ADD     A,B                     ;
        RETURN                          ;
        .END
```

The patch source is assembled using MACRO-11 and the resulting object file is input to PAT along with the original object file. The result of PAT processing would then appear as follows:

```
        .TITLE  ABC
        .IDENT  /01.01/
ABC::
        MOV     A,C                     ;
        CALL    XYZ                     ;
        RETURN                          ;
.=ABC
.=.+12
        ADD     A,B                     ;
        CALL    XYZ                     ;
        RETURN                          ;
        .END
```

After Task Builder processes these files, the task image appears as follows:

```
        .TITLE  ABC
        .IDENT  /01.01/
ABC::
        MOV     A,C                     ;
        CALL    XYZ                     ;
        ADD     A,B                     ;
        RETURN                  ;
        .END
```

The Task Builder uses the .=.+12 in the program counter field to determine where to begin overlaying instructions in the program. The Task Builder overlays the RETURN instruction with the patch code:

```
        ADD     A,B
        RETURN
```

Adding a Subroutine to a Module

The second example illustrates a technique for adding a subroutine to an object module. In many cases, a patch requires that more than a few lines be added to patch the file. A convenient technique for adding new code is to append it to the end of the module in the form of a subroutine. This way, you can insert a CALL instruction to the subroutine at an appropriate location. The CALL directs the program to branch to the new code, execute that code, and then return to in-line processing. The input file for the example is:

```
        .TITLE  ABC
        .IDENT  /Ø1/
ABC::   MOV     A,B             ;
        CALL    XYZ             ;
        MOV     C,RØ            ;
        RETURN                  ;
         *
         *
         *
        .END
```

The correction file for this example is as follows:

```
        .TITLE  ABC
        .IDENT  /Ø1.Ø1/
        CALL    PATCH           ;
        NOP                     ;
        .PSECT  PATCH
PATCH:
        MOV     A,B             ;
        MOV     D,RØ            ;
        ASL     RØ              ;
        RETURN                  ;
        .END
```

PAT appends the correction file to the input file, as in the previous example. The Task Builder then processes the file and the following output file is generated:

```
        .TITLE  ABC
        .IDENT  /Ø1.Ø1/
ABC::
        CALL    PATCH           ;
        NOP
        CALL    XYZ             ;
        MOV     C,RØ            ;
        RETURN                  ;
         *
         *
         *
        .PSECT  PATCH
PATCH:
        MOV     A,B             ;
        MOV     D,RØ            ;
        ASL     RØ              ;
        RETURN                  ;
        .END
```

In this example, the CALL PATCH and NOP instructions overlay the three-word MOV A,B instruction. (The NOP is included because this is a case where a two-word instruction replaces a three-word instruction and NOP is required to maintain alignment.) The Task Builder allocates additional storage for PSECT PATCH, writes the specified code into this program section, and binds the CALL instruction to the first address in this section. Note that the MOV A,B instruction replaced by the CALL PATCH is the first instruction executed by the PATCH subroutine.

A.3.5 Determining and Validating the Contents of a File

Use the Checksum switch (/CS) to determine or validate the contents of a module. The Checksum switch directs PAT to compute the sum of all binary data comprising a file. If specified in the form /CS:number, /CS directs PAT to compute the checksum and compare that checksum to the value specified as number.

To determine the checksum of a file, enter the PAT command line with the /CS switch applied to the file whose checksum is being determined, for example:

```
=INFILE/CS,INFILE.PAT
```

PAT responds to this command with the message:

```
INPUT MODULE CHECKSUM IS <checksum>
```

A similar message is generated when the checksum for the correction file is requested.

To validate the size of a file, enter the Checksum switch in the form /CS:number. PAT compares the value it computes for the checksum with the value you specify as number. If the two values do not match, PAT displays a message reporting the checksum error:

```
ERROR IN FILE <filename> CHECKSUM
```

Checksum processing always results in a nonzero value.

A.4 PATCH MESSAGES

PAT generates messages that state checksum values and messages that describe error conditions. For checksum values and nonfatal error messages, PAT prefixes messages with:

```
PAT -- *DIAG -
```

For error messages that describe errors that caused PAT to terminate, PAT uses the prefix:

```
PAT -- *FATAL -
```

The messages described below are grouped according to message type, as follows:

- Information Messages.
- Command line errors.
- Input/Output errors.
- Errors in file contents or format.
- Internal software errors.
- Storage allocation errors.

A.4.1 Information Messages

The following messages describe results of checksum processing.

CORRECTION INPUT FILE CHECKSUM IS <checksum>

> Description: <Checksum> is the module checksum printed in response to the /CS switch appended to a correction input file specification. The value is an octal quantity.
>
> Suggested User Response: No response necessary.

INPUT MODULE CHECKSUM IS <checksum>

> Description: <Checksum> is the module checksum printed in response to the /CS switch appended to an input file specification. The value is an octal quantity.
>
> Suggested User Response: No response necessary.

A.4.2 Error Conditions

The following errors result from failure to adhere to the command line syntax rules.

COMMAND LINE ERROR <command line>

COMMAND SYNTAX ERROR <command line>

> Description: The command line displayed contains an error.
>
> Suggested User Response: Reenter the command line using the correct syntax.

ILLEGAL INDIRECT FILE SPECIFICATION <command line>

> Description: The printed command line contains an indirect file specification that contains one of the following errors:
>
> - A syntax error in the file specification.
> - Specification of a non-existent indirect file.
>
> Suggested User Response: Check for file specification syntax errors. Ensure that the specified file is contained in the User File Directory.

MAXIMUM INDIRECT FILE DEPTH EXCEEDED <command line>

> Description: The command line displayed specifies an indirect file that exceeds the nesting level permitted by the PAT Utility (2).
>
> Suggested User Response: Reorder your files so that they do not exceed the PAT utility nesting limit (2).

A.4.3 File Specification Errors

The following messages are caused by errors in the specification of input or output files or related file switches.

CORRECTION INPUT FILE MISSING <command line>

> Description: The command line does not contain the mandatory correction file input specification.
>
> Suggested User Response: Reenter the command line specification including the correction file.

ILLEGAL DEVICE/VOLUME SPECIFIED <device name>

> Description: <device name> does not adhere to the syntax rule for specifying device or volume.
>
> Suggested User Response: Check the rules for specifying the device and reenter the command line.

ILLEGAL DIRECTORY SPECIFICATION <directory name>

> Description: The directory string displayed does not adhere to the syntax rules for specifying directories.
>
> Suggested User Response: Reenter the command line specifying the directory string in the correct syntax.

ILLEGAL FILE SPECIFICATION <filename>

> Description: The filename printed does not adhere to the syntax rules for file specifications.
>
> Suggested User Response: Reenter the command line using the correct syntax for the filename.

ILLEGAL SWITCH SPECIFIED <filename>

> Description: An unrecognized switch or switch value has been appended to the filename displayed.
>
> Suggested User Response: Check the rules for specifying the switch and reenter the command line.

INVALID FILE SPECIFIED <filename>

> Description: The filename displayed is associated with one of the following error conditions:
>
> - Nonexistent device.
> - Nonexistent directory - filename is the name of the User File Directory associated with the file to be processed.
>
> Suggested User Response: Correct the device or directory specification and reenter the command line.

MULTIPLE OUTPUT FILES SPECIFIED <command line>

Description: Only one output file specification is accepted.

Suggested User Response: Reenter the command line with only one output file specified.

REQUIRED INPUT FILE MISSING <command line>

Description: The command line does not contain the mandatory input file specification.

Suggested User Response: Reenter the command line specifying the input file.

TOO MANY INPUT FILES SPECIFIED <command line>

Description: The command line displayed contains too many input file specifications. PAT only accepts the input and correction file specifications.

Suggested User Response: Reenter the command line specifying the correct files.

UNABLE TO FIND FILE <filename>

Description: The specified input or correction file could not be located.

Suggested User Response: Ensure that the file exists. Reenter the command line.

A.4.4 Input/Output Errors

The error messages listed below are caused by faults detected while performing I/O to the specified file.

ERROR DURING CLOSE: FILE: <filename>

Description: This error is most likely to occur while attempting to write the remaining data into the output file before de-accessing it. The principal sources of error under these circumstances are:

- Device full.
- Device write-locked.
- Hardware error.

Suggested User Response: Perform the appropriate corrective action and reenter the command line. If the problem appears to be a hardware problem rather than faulty media, contact your DIGITAL Field Service Representative.

ERROR POSITIONING FILE <filename>

Description: An attempt has been made to position the file beyond end-of-file.

Suggested User Response: submit a Software Performance Report along with the related console dialogue and any other pertinent information.

I/O ERROR ON INPUT FILE <filename>

Description: An error was detected while attempting to read the specified input file. The principal cause is a device hardware error.

Suggested User Response: Reenter the command. If the problem persists, submit a Software Performance Report along with the related console dialogue and any other pertinent information.

I/O ERROR ON OUTPUT FILE <filename>

Description: An error occurred while attempting to write into the named output file. The most likely causes are:

- Device full.
- Device write-locked.
- Hardware error from device.

Suggested User Response: Perform the appropriate corrective action and reenter the command. If the problem appears to be a hardware problem rather than faulty media, notify your DIGITAL Field Service Representative.

A.4.5 Errors in File Contents or Format

The following errors represent inconsistencies detected by PAT in the format or contents of the input or correction files.

ERROR IN FILE <filename> CHECKSUM

Description: Checksum computed by PAT for the named file does not match that supplied by the user.

Suggested User Response: Ensure that the correct checksum was specified. If the checksum was correct, an invalid version of the file was specified on the command line. Rerun PAT specifying the correct version of the file.

FILE <filename> HAS ILLEGAL FORMAT

Description: The format of the named file is not compatible with the object files produced by the standard DIGITAL language processors or accepted by the Task Builder. The principal causes are:

- Truncated input file.
- Input file consists of text.

Suggested User Response: Ensure that the file is in the correct format and resubmit it for PAT processing.

INCOMPATIBLE REFERENCE TO GLOBAL SYMBOL <symbol name>

Description: Correction input file contains a global symbol whose attributes do not match one or more of the following input file symbol attributes:

- Definition or Reference
- Relocatable or Absolute

Suggested User Response: Update the correction input file by modifying the symbol attributes. Reassemble the file and resubmit it for PAT processing.

INCOMPATIBLE REFERENCE TO PROGRAM SECTION <section name>

Description: Correction input file contains a section name whose attributes do not match one or more of the following input file section attributes:

- Relocatable or Absolute
- .PSECT or .CSECT

Suggested User Response: Update the correction file by modifying the section attribute or changing the section type. Reassemble the file and resubmit it to PAT for processing.

UNABLE TO LOCATE MODULE <module name>

Description: The module name specified in the correction input file could not be found in the file of concatenated input modules.

Suggested User Response: Update the input file specification to include the missing module. Reenter the command line.

A.4.6 INTERNAL SOFTWARE ERROR

These errors reflect internal software error conditions.

ILLEGAL ERROR-SEVERITY CODE <error data>

Description: An error message call has been generated containing an illegal parameter.

Suggested User Action: if they persist, submit a Software Performance Report along with the related console dialogue and any other pertinent information.

A.4.7 STORAGE ALLOCATION ERROR

The following error message indicates that insufficient task memory was available for storing global symbol and program section data.

NO DYNAMIC STORAGE AVAILABLE <storage-listhead>

Description: Insufficient contiguous task memory was available to satisfy a request for the allocation of storage.

<Storage-listhead> is a display of the two-word dynamic storage listhead contents in octal.

Suggested User Response: If possible, PAT should be reinstalled with a larger increment.

PART 2
COMMAND SPECIFICATIONS

PART 2

COMMAND SPECIFICATIONS

COMMAND FORMAT

The general format of a command is:

```
     [$]command-name[qualifiers] [parameter-1][,...,parameter-n]
```

The following rules apply:

1. Brackets - In the description of commands in this manual, brackets ([and]) are used to surround optional values. For example:

   ```
   COPY[qualifiers]
   ```

   indicates that the user does not need to supply any qualifiers to issue a valid COPY command.

2. Dollar Sign ($) - The dollar-sign ($) must appear in position 1 of a command to be executed in batch mode. It may optionally appear in a command executed in interactive mode.

3. Command-Names - The command name describes the action the command is to perform. With the exception of LOGIN, LOGOUT, DEASSIGN and DEALLOCATE, which can be abbreviated to 4 letters, all commands can be abbreviated to 3 letters or fewer. Additional letters are acceptable, for example, LOGOUT, LOGOU and LOGO are all correct.

4. Parameters - A parameter either describes a value that a command is to use when executing or it further defines the action a command is to take. Interactive users may supply parameters in response to prompts (see Chapter 4, Section 4.1). Otherwise, at least one space must separate the first parameter from the command-name; parameters are then separated from each other by one or more spaces and/or a single comma (,).

5. Parentheses and Ellipses - Some commands permit the user to replace a single parameter by a list of values. When this is done the list may be surrounded by parentheses. Parentheses are not required when the parameter being replaced is the only or the last parameter in the command string.

Examples:

a. `DELETE (A.DAT;2, B.DAT;1, C.DAT;4)`

The parentheses are optional

b. `APPEND (A.DAT B.DAT C.DAT) D.DAT`

The parentheses are required because the parameter being replaced is not the last parameter. (This command specifies that files A.DAT, B.DAT and C.DAT are to be added to the end of file D.DAT).

In the description of a command's format, ellipses (three dots "...") indicate that a list of values of the same type may replace a single value.

6. Qualifiers - A qualifier is used to modify the default action of a command. There are defaults for most qualifiers. A qualifier always begins with a slash (/). Both command names and parameters can be qualified.

   Examples:

   ```
   PRINT/DELETE   MYFILE.DAT

   CREATE   DAT36.DAT/PROTECT:(WO:RWED)
   ```

   Many qualifiers have associated qualifier values. The qualifier is separated from the qualifier value by a colon (:), e.g. KEEP:1. Whenever a qualifier requires a list of values, that list must be enclosed in parentheses, e.g.

   ```
   /BLOCKS:(m-n)
   ```

   A qualifier may not contain any spaces.

7. Continuation Character (-) - A hyphen (-), which may be optionally followed by spaces and/or a comment, is used to indicate that a command is to be continued on the next line.

   Example:

   ```
   PDS> COPY A.DAT -

   >B.DAT
   ```

   Note that following a continuation character, the system reprompts with a prompt sign (>) on the following line.

8. Comment Character (!) - An exclamation mark delimits the start of a comment. Comments can occur only after the last character of a command or after a hyphen. Comments are for the user's information only and do not affect the processing of the command.

Examples:

```
PDS> COPY A.DAT  B.DAT  !FILE A TO FILE B.

PDS> MOUNT/DENSITY:800 MT: - !  MOUNT MY
>    VOLID3 TU10:            ! TAPE ON ANY TU10
```

9. Concatenation Character (+) - A plus sign (+) indicates concatenation, that is, the records in the file specified on the left of the plus sign are processed followed by the records in the file specified on the right of the plus sign.

   Example:

   ```
   MACRO      A+B
   ```

   The MACRO-11 statements in file A.MAC followed by the MACRO-11 statements in file B.MAC are read by the MACRO-11 assembler.

   See Chapter 4 for further details on issuing PDS commands.

DICTIONARY OF PDS COMMANDS

The following PDS commands are specified in this section.

```
ABORT                      EDIT                       MERGE
ALLOCATE                   ENABLE                     MESSAGE
APPEND                     $EOD                       MOUNT
ASSIGN                     $EOJ
                                                      ON
BASIC                      FIX
                           FORTRAN                    PRINT

CANCEL                     GOTO                       QUEUE
COBOL
COMPARE                    HELP                       REMOVE
CONTINUE                                              RENAME
COPY                       INITIALIZE                 RUN
CORAL                      INSTALL
CREATE                                                SET
                           $JOB                       SHOW
DEALLOCATE                                            SORT
DEASSIGN                   LIBRARIAN                  STOP
DELETE                     LINK                       SUBMIT
DIRECTORY                  LOGIN
DISABLE                    LOGOUT                     TYPE
DISMOUNT                                              UNFIX
DUMP                       MACRO                      UNLOCK
```

ABORT

The ABORT command can abort the execution of the user's current timesharing task. The timesharing task must previously have been suspended, for example by typing CTRL/C.

The user can also abort a real-time task that is running for either the current terminal or a specified terminal.

FORMAT 1

```
    PDS> ABORT
```

DESCRIPTION

This format aborts the timesharing task controlled from the user's terminal and currently suspended.

EXAMPLE

```
    CTRL/C
    TASK SUSPENDED
    PDS> ABORT
    PDS>
```

FORMAT 2

```
    PDS> ABORT/REALTIME
    TASK?  taskname
    [TERMINAL?  terminal]
```

where

taskname	is the installed name of the task to be aborted
terminal	is the terminal from which the task to be aborted was activated. The default is the current terminal.

EXAMPLES

```
    PDS> ABORT/REALTIME RTTSK

    PDS> ABORT/REALTIME MYTSK TT6
```

ALLOCATE

The ALLOCATE command allocates a specified device to the user and optionally associates a logical name with the device.

FORMAT

```
     PDS> ALLOCATE

     RESOURCE?  DEVICE

     DEVICE?  device-name

     LOGICAL NAME?  logical-name
```

or

```
     $ALLOCATE DEVICE device-name logical-name
```

where

device-name	is the specification of the device to be allocated to the user.
logical-name	is a logical name to be associated with the device, of the form XYn or XYmn. X, Y are alphabetic, m and n octal digits. At least one digit must be specified, even if Ø.

DESCRIPTION

The user has exclusive access to the device until either it is explicitly deallocated or until the system deallocates it. The system automatically deallocates when the user dismounts the device or deassigns the last logical unit number to which the device is assigned, unless the user modifies the DISMOUNT or DEASSIGN command with the qualifier /KEEP.

The user may not explicity allocate a system device, that is, a device allocated to all users by the system manager. If device-name does not include a unit number, the system allocates any available device of the specified type and, in interactive mode, prints at the user's terminal the physical unit allocated. In this case, the batch user must define a logical name in order to refer to that device in subsequent commands.

EXAMPLES

1.
```
PDS> ALLOCATE
RESOURCE? DEVICE
DEVICE? MT: <ALT>
LOGICAL NAME? MYØ:
PDS> MOUNT MYØ:
VOLUME-ID? VOL75

PDS> DISMOUNT/KEEP
DEVICE? MYØ
VOLUME-ID? VOL75
PDS> MOUNT MYØ: VOL73
PDS> DISMOUNT
DEVICE? MYØ
VOLUME-ID? VOL73
```

2.
```
$ALLOCATE DEVICE MT:  LMØ
```

3.
```
ALLOCATE DEV DKØ:
```

APPEND

The APPEND command appends records from one or more input files, to the end of an already existing SEQUENTIAL file. The input file list can be either a list of SEQUENTIAL files, named explicitly or using wildcards, or a single INDEXED or RELATIVE file whose name has been supplied explicitly. When more than one input file is specified, the files are appended in the order in which they appear in the commands.

FORMAT

```
    PDS> APPEND

    FILE?  [(]infile-1[,...infile-n)][/qualifier][)]

    TO? outfile
or

    $APPEND [(]infile-1[,...infile-n)][/qualifier][)] outfile
```

where

infile	is an input file specification
outfile	is the file specification to be updated

/qualifier is one of:

/SEQUENTIAL

input file is sequential (default)

/INDEXED[/KEY:NUMBER:n]

input file is an Indexed Sequential (ISAM) file. The order in which records are appended can be specified by the /KEY qualifier and key number.

Default: /KEY:NUMBER:1 (the primary key).

/INDEXED may be omitted if /KEY:NUMBER:n is specified.

/RELATIVE

input file is RELATIVE structured.

DESCRIPTION

If one or more files in a list of input files is in error, the system ignores the files in error and appends the rest to the output file.

All file specifications must include a filename and a filetype.

If a version number is not specified, the system assumes the highest version number for the input file, and the highest version plus 1 for the output file.

If one of the specified files is to be input from the user's terminal (TI:), the system transfers all that the user types in after the completed command string. The transfer continues until the user types CTRL/Z to terminate the input file. Such input from TI: is sequential by nature.

EXAMPLES

1. `PDS> APPEND (A.OBJ B.OBJ) C.OBJ`

2.
   ```
   PDS> APPEND
   FILE? (ABC.FTN DEF.FTN)
   TO?  XYZ.FTN
   ```
3. `PDS> APPEND TWO.MAC, ONE.MAC`

4. `$APPEND (ABC.DAT,DEF.DAT), XYZ.DAT`

5. `PDS> APPEND ADDIT.DAT/KEY:NUM:3 OLDONE.DAT`

   Appends all records from the ISAM file ADDIT.DAT to OLDONE.SAV in an order determined by key number 3 (the second alternate key field).

6. `PDS> APPEND FILE1.TXT+[200,40]*.TXT UPDATED.TXT`

   Appends text file FILE1 and all .TXT files in [200,40] to UPDATED.TXT in the current UFD.

ASSIGN

The ASSIGN command assigns a device to a logical unit.

The assignment can only apply to the timesharing task(s) that will be run from the terminal (FORMAT 1) or to a named installed task (FORMAT 2).

Assignment does not affect any currently active version of a task.

To determine the current assignments of LUNs for an installed task use the command SHOW LUNS taskname.

FORMAT 1 (timesharing)

```
    PDS> ASSIGN

    FILE?  device-name

    LUN?  lun
```

or

```
    $ASSIGN device-name lun
```

where

device-name	is the specification of the device to be assigned to the logical unit. The device must be one allocated to the user by the ALLOCATE or MOUNT command, or one to which all users have access.
lun	is a logical unit number.

EXAMPLES

```
     1.  $ASSIGN DPØ:   7

     2.  PDS> ASSIGN

         FILE?  LPØ:

         LUN?  6

     3.  PDS> ASSIGN DK2:

         LUN?  5
```

FORMAT 2 (ASSIGN/REALTIME)

```
     PDS> ASSIGN/REALTIME
     TASK?  taskname
     DEVICE?  device-name
     LUN?  lun
```

where

taskname	is the installed name of the task for which the installed LUN assignment is to be changed
device-name	is the device for which the LUN is to be assigned
lun	is the LUN to be associated with the specified device

EXAMPLE

```
PDS> ASSIGN/REAL MART DK2:  3
```

For the installed task MART re-assign LUN3 to device DK2.

DESCRIPTION

Users may assign logical unit numbers at three points:

1. By means of the ASSIGN command before the user runs a task.
2. By means of a Task Builder option when a task is linked (see the IAS Task Builder Reference Manual).
3. From within a program by means of the system directive ALUN$ or OPEN$ or the FORTRAN subroutines ASSIGN and ASNLUN (see the IAS Executive Reference Manual- Volume I).

If the ASSIGN command associates a device name with a logical unit number, that assignment overrides any made for that logical unit number by the Task Builder. And if an executing program assigns a logical unit, that assignment overrides the action of any ASSIGN command for that logical unit number. The system automatically deassigns logical units when the associated volume is dismounted or device deallocated. The user may also issue the DEASSIGN command to deassign a device from a logical unit.

BASIC

The BASIC command invokes a BASIC language processor which may be either the BASIC-11 interpreter or the BASIC-11 interpreter or the BASIC-PLUS-2 compiler.

FORMAT

```
     BASIC [qualifier]
```

where

qualifier is one of the following command qualifiers:

Qualifier	Meaning
/B11	Invoke the BASIC-11 interpreter. Applicable to systems that have both BASIC-11 and BASIC-PLUS-2. If omitted, the system invokes the installation's default BASIC.
/BP2	Invoke the BASIC-PLUS-2 compiler. Applicable to systems that have both BASIC-11 and BASIC-PLUS-2. If omitted, the system invokes the installation's default BASIC.

The BASIC command has no parameters.

DESCRIPTION

The following description relates to the BASIC-11 interpreter only. For details of BASIC-PLUS-2 see BASIC-PLUS-2 specific documents.

When the user issues the BASIC command, BASIC indicates that the interpreter is ready to receive a command or program line by typing "READY".

To terminate a BASIC session and return control to PDS, the user types 'BYE' on a new line. The system then prints information about the session and prompts for further PDS commands. For example:

```
     BYE


     15:57:32  SIZE: 1ØK  CPU: 3:Ø9


     PDS>
```

Effect of CTRL/C

When the BASIC interpreter is executing a program, CTRL/C causes the system to stop execution after the current line. The terminal displays the number of the last line executed and the user may then issue further BASIC commands.

CTRL/C typed during the execution of a BASIC LIST or SAVE command or an immediate mode statement stops the execution of those commands or statements. It has no effect on the execution of other BASIC commands.

CANCEL

The CANCEL command allows the user to cancel periodic scheduling of requests for a real-time task.

FORMAT

```
     PDS> CANCEL
     TASK?  taskname
     [TERMINAL?  terminal]
```

where

taskname	is the installed name of the task whose scheduled requests are being cancelled
terminal	is the terminal from which the task to be cancelled was activated. The default is the user's terminal.

EXAMPLES

```
     PDS> CANCEL XKE2

     PDS> CAN MYTS TT4
```

COBOL

The COBOL command compiles a COBOL source program.

FORMAT

```
     PDS> COBOL[qualifiers]

     FILE?  filespec
```

or

```
     $COBOL[qualifiers] filespec
```

where

filespec	is a specification of the file containing the COBOL source program. The specification must contain a filename. If the filetype is omitted, the system assumes it to be CBL.
qualifiers	are one or more of the following:

Qualifier	Meaning
/OBJECT[:filespec]	Produce an object file, named according to filespec if it is supplied (no wild-cards allowed). The default filetype is OBJ. /OBJECT is the default qualifier.
/NOOBJECT	Do not produce an object file.
/LIST[:filespec]	Produce a listing file named according to filespec if it is supplied (no wild-cards allowed). The default filetype is .LST.
/NOLIST	Do not produce a listing file (the default condition for interactive mode).
/SWITCHES:(switches)	Apply the specified COBOL compiler switches. See the section called Compiler Switches below.

Defaults

Object File - If the qualifier /OBJECT is specified without a file specification, the object file is given the name of the source file and the filetype .OBJ. The system default is /OBJECT.

Listing File - If /LIST is supplied without a filespec the listing file is sent to the line printer. /NOLIST is the system default.

Compiler Switches

The COBOL command includes compiler switches that permit the user to tailor the compilation listing to meet particular needs.

A list of switches and their meanings follow:

Switch	Meaning	Default
/HELP	Display on the user terminal information about how to use the compiler switches.	
/ERR:n	Suppress the printing of diagnostics with a severity number of less than n. The range of n must be Ø through 2. where: Ø = Informational diagnostics 1 = Warning diagnostics 2 = Fatal diagnostics The switch cannot suppress severity 2 (fatal) diagnostics. (An entry of 2 suppresses the printing of all severity numbers that are less than 2.)	/ERR:Ø
/ACC:n	Produce an object program only if the source program contains diagnostics with severities equal to or less than n. The range of n must be Ø through 2.	/ACC:1
/MAP	Produce special map listings of Data Division Procedure Map External Subprograms Referenced Data and Control PSECTs OTS Routines Referenced Segmentation Map	
/NL	Instruct the compiler not to list the source statements copied from a library file. The resultant source listing contains only the COPY statement.	
/CVF	The source program is in conventional format (i.e., 8Ø-character source lines with Area A beginning in character position 8. The default is that area A begins at position 1).	
/CREF	Include a cross-reference listing as a part of the listing file output. When /CREF is specified, data-names, procedure-names, and source line numbers are sorted into ascending order and appended to the end of the compilation listing. The symbol # is used to indicate those lines that contain the lines in which the reference name is defined.	

Switch	Meaning	Default
	NOTE The use of /CREF significantly slows down the compilation of large programs.	
/CSEG:nnnn	Allows you to specify the maximum size procedural code PSECT to be produced by the compiler where nnnn is the maximum size procedural code PSECT, in decimal bytes. The minimum value of nnnn is 1ØØ.	
/KER:kk	Instruct the compiler to generate PSECT names using the two-character kernel specified by kk to make them unique to this compilation, where kk is a two character string that may contain the numbers Ø through 9 and the letters A through Z.	
/OBJ	Print the object location in which the code for each verb of the program is located. The information is listed on the linee proceeding the source statement it describes.	
/ODL	Generate an ODL file (default condition). To override the default condition, enter /-ODL.	
/OV	Make all procedural PSECTs (segments) overlayable. Therefore, the root or main program will contain no procedural statements.	
/PFM:nn	Define the maximum number of nested PERFORM statements in the program being compiled. If specified, the compiler generates a nested PERFORM stack equal in depth to the decimal number specified by nn. The default nested perform size is 1Ø. It is to your advantage to use this switch to adjust the nested PERFORM stack size to the exact number required. This assures maximum utilization of memory in that only the exact amount of PERFORM stack space is generated.	
/PLT	Automatically pool literals to minimize the memory required to store them (default condition). Pooling of literals, however, slows down compiler execution speed. To bypass literal pooling for increased compiler speed, enter /-PLT.	
/RO	Generate read-only PSECTs for the Procedure Division object modules.	
/SYM:n	Obtain more symbol table space for the compilation. "n" (an integer in the range of 1 through 4) specifies the space required for the maximum number of data-names and procedure-names allowed in the compilation. See Table P2-1 for the correspondence between the integer specified by n, and the number of data-names and procedure-names assigned.	

Table P2-1
/SYM:n Switch Values

n	Maximum Data-Names	Maximum Procedure-Names
1	761	761 (default)
2	1Ø21	1Ø21
3	1531	1531
4	2Ø39	2Ø39

EXAMPLES

1.
```
PDS> COBOL COBPROG.CBL
```

2.
```
PDS> COBOL/SWITCHES:(/MAP)
FILE? COBPROG.CBL
```

3.
```
$COBOL BATCHCOB
```

COMPARE

The COMPARE command is used to compare two files line by line with one another and produce one of:

1. A listing of the differences found

2. A listing of one file with the differences flagged

3. A SLIPER file that converts one file to the other.

FORMAT

```
PDS> COMPARE/qualifiers oldfile newfile
```

where

qualifiers are any of the following:

/[NO]OUTPUT[:filespec]
to output all the differences found to the lineprinter. If a file specification is given the output will be directed to the specified file. If /NOOUTPUT is specified, only the number of differences found will be printed.

Default:/OUTPUT

/CHANGE_BARS[:n]
n is the decimal character code to be used. 'newfile' is printed with those lines which differ from 'oldfile', marked by the specified character. For example, 124 for vertical bar (octal 174).

Default: decimal code 33 for exclamation point (!) (octal 41).

/[NO]COMMENT
to include all comments in the file comparison. If /NOCOMMENT is specified, all comments will be ignored.

Default: /COMMENT

/[NO]FORM_FEEDS
to include all form feeds in the file comparison. If /NOFORM_FEEDS is specified, records containing only a formfeed will be ignored.

Default: /NOFORM_FEEDS

/LINES:n
to specify the number of lines that determine a match. This match means that n successive lines in each input file have been found identical. When a match is found, all differences occurring before the match are output. In addition, the first line of the current match is output after the differences to aid in locating the place within each file at which the differences occurred.

Default: 3 lines

/[NO]MULTIPLE_BLANKS
to include all multiple blanks (that is, spaces and tabs) in the file comparison. If /NOMULTIPLE_BLANKS is specified, all multiple blanks will be ignored.

Default: /MULTIPLE_BLANKS

/[NO]TRAILING_BLANKS
to include all trailing blanks in the file comparison. If /NOTRAILING_BLANKS is specified, all trailing blanks will be ignored.

Default: /TRAILING_BLANKS

/SLIPER
Output a SLIPER file that converts oldfile to newfile.

Default: /NOSLIPER

/[NO]BLANK_LINES
to include all blank lines in the file comparison. If /NOBLANK_LINES is specified, all blank lines will be ignored.

Default: /NOBLANK_LINES

oldfile	is the old file to be used in the file comparison.
newfile	is the new file to be compared with the old file.

EXAMPLE

1. <u>PDS></u> COMPARE/NOOUTPUT/FORM_FEEDS/NOMULT

   <u>OLDFILE?</u> MKXØ3.MAC;1

   <u>NEWFILE?</u> MKXØ3.MAC

2. <u>PDS></u> COMPARE/SLIPER BCPLIO.MAC;1

   <u>NEWFILE?</u> BCPLIO.MAC;Ø

CONTINUE

The CONTINUE command has four formats. In interactive timesharing the CONTINUE command causes the currently suspended user task to resume execution. See FORMAT 1.

In an indirect command or batch command file, CONTINUE has no effect other than proceeding to the next command. It does not imply previous suspension of a task. See FORMAT 2.

In real time, the CONTINUE command is used:

1. To continue the execution of a task after it has been suspended using the 'suspend' form of message output (CONTINUE/MESSAGE). See the IAS/RSX-11D Device Handlers Reference Manual, Chapter 11. See FORMAT 3.

2. To resume execution of a previously suspended task, after being suspended by the SUSPEND (SPND$) Directive (CONTINUE/REALTIME). See FORMAT 4.

FORMAT 1

```
PDS> CONTINUE
```

DESCRIPTION

In interactive timesharing, the CONTINUE command may only be issued after the user task has been suspended by typing CTRL/C. Typing CONTINUE reactivates the currently suspended task.

EXAMPLE

```
CTRL/C

TASK SUSPENDED

PDS> CONTINUE
```

FORMAT 2 (indirect or batch)

```
[$]CONTINUE
```

DESCRIPTION

In an indirect command or batch command file, CONTINUE has no effect other than proceeding to the next command. It does not imply previous suspension of a task.

FORMAT 3

```
PDS> CONTINUE/MESSAGE
TASK?  taskname
[TERMINAL?  terminal]
```

where

taskname	is the installed name of the task to be continued after being suspended by the 'suspend' form of message output (that is, the MO message handler).
terminal	is the terminal from which the task to be resumed was activated. The default is the user's terminal.

FORMAT 4

```
PDS> CONTINUE/REALTIME
TASK?  taskname
[TERMINAL?  terminal]
```

where

taskname	is the installed name of the task being "resumed" after previously being suspended by the SUSPEND directive.
terminal	is the terminal from which the task to be resumed was activated. The default is the user's terminal.

EXAMPLES

```
PDS> CONTINUE/MESSAGE MYTTSK

PDS> CONT/REALTIME XKEE3 TT2
```

COPY

The COPY command copies:

1. one sequential file to another sequential file;
2. using wildcards, a group of sequential files to another group of sequential files;
3. the concatenation of a number of sequential files to a single sequential file.
4. records from a single INDEXED or RELATIVE file to a single sequential file. For making a copy of an INDEXED or RELATIVE file to a file of the same kind, see the MERGE command.

FORMAT

```
     PDS> COPY[qualifiers]

     FROM? infile[file-qualifier]

     TO?  outfile[file-qualifier]
```

or

```
     $COPY[qualifiers]  infile[file-qualifier], outfile[file-qualifier]
```

where

infile — is an input file specification. Concatenated files are linked by a plus sign (+).

For example:

```
filespec+filespec+filespec+....
```

outfile — is an output file specification.

qualifiers — are one or more of the following:

Qualifier	Meaning
/ALLOCATION:n	Allocate n blocks to the output file
/CONTIGUOUS	Make the output file contiguous. Note that this qualifier has no effect when copying from magnetic tape.
/OWN	Make the destination UFD the owner of the copy or copies. Not applicable to foreign files (see under file-qualifier, below).
/REPLACE	Replace the existing output file, if any.

Qualifier	Meaning
/ASCII[:n]	Formatted ASCII (for Foreign files) The transfer is performed as formatted ASCII. Formatted ASCII is defined as ASCII data records terminated by a carriage return or a form feed. If n is specified, fixed length records of size n are generated. Output records will be padded with nulls, if necessary. If n is not specified, then variable length records are generated. The output record size equals the input record size.

NOTE

ASCII data is transferred as 7-bit quantities. The eighth bit of each byte is masked off before transfer. CTRL/Z (ASCII Ø32(8)) is treated as logical end of input file for formatted ASCII transfers from DOS-11 cassette to Files-11.

[NULLs (ASCII ØØØ(8)), RUBOUTs (ASCII 177(8)) and Vertical Tabs (ASCII Ø13(8)) are ignored.]

/BINARY[:n]	Formatted Binary (for Foreign files) The output file is to be formatted binary. If n is specified, n indicates the fixed length size record in bytes (512 bytes is the maximum). The command pads records with nulls to create the specified length. If n is not specified, standard DOS and RT-11 formatted binary records are produced.
/IMAGE[:n]	Image Mode (for Foreign files) The output file is to be in image mode. Image mode forces fixed length records. n can be used to indicate the desired record length (512 bytes is the maximum) or if n is not specified, then 512 bytes is assumed.
/BLOCKSIZE:n	Specifies the block size for cassette tape output. n = the block size in bytes. If /BLOCKSIZE is not specified, a block size of 128 is assumed. /BLOCKSIZE is valid only in a CT output file specifier.

Qualifier	Meaning
/VERIFY	/VERIFY is valid only with a CT output file specifier. Verify after write -- Causes each record written to the cassette, to be read and verified.

/SEQUENTIAL (Input file only)
Input file is a sequential file.

/INDEXED[/KEY:NUMBER:n] (Input file only)

The single input file is an Indexed Sequential (ISAM) file.

The records from the (implied /INDEXED) input file are to be copied in the order determined by key number n (n>Ø) to create a new sequential file.

If /INDEXED is specified but not /KEY, the records are copied in the order determined by key number 1 (the primary key).

/RELATIVE (Input file only)

The single input file has RELATIVE organization.

If none of the last four qualifiers is used, /SEQUENTIAL is assumed by default.

file-qualifier modifies the specification of a foreign file in DIGITAL'S DOS or RT-11 format. The qualifiers are:

/DOS
/RT11

DESCRIPTION

If infile or outfile has a filename then it must also have a filetype.

If the version number is omitted from the input file then the highest version number is used. If it is omitted from the output file then the highest version number plus 1 is used.

Wild-cards are allowed whenever an input file specification does not describe concatenated files. If any of the filename, filetype or version fields of the output file contain a wild-card, all fields must be wild; the version field, however, may be omitted. If one part of the output file UFD is a wild-card, both parts must be wild.

If a 'wild' version number is specified in an output file specification, the version numbers for that file will be preserved. If a filename is not specified the system assumes wild-cards (that is, *.*;*).

If /DOS or /RT11 modifies either file specification, then the input files may not be concatenated and the output filename and filetype must be wild (that is, foreign files may not be renamed).

If the user enters infile from the user's terminal (TI:), the system transfers to the output file all that the user types in after the completed command string. The transfer continues until the user types CTRL/Z to terminate the input file.

If either infile or outfile is not in Files-11 format, its specification must be modified by either /DOS or /RT11. The system does not accept any other foreign formats.

Because of the unused space at the end of blocks, if a file is copied from disk to magnetic tape it will occupy more blocks on the tape than it did on the disk. Furthermore, when the file is copied from magnetic tape back to disk, the resulting disk file is also longer than the original disk file.

The COPY command is not the best method of making a replica of an Indexed Sequential file. A preferable method, which will also tidy the internal structure of the file, is to create a new, empty Indexed Sequential file having the same structure as the original file (using the CREATE command) and then merge the original file into the new (empty) file (using the MERGE command).

EXAMPLES

```
1.   PDS> COPY A.CBL B.CBL

2.   $COPY E.TXT, F.TXT

3.   PDS> COPY
     FROM?  E.TXT
     TO?  F.TXT

4.   PDS> COPY/OWN DK0:[*,*]*.*
     TO? DK1:[*,*]*.*

5.   PDS> COPY DATA.DAT DT0:*.*

6.   PDS> COPY/IMAGE:100/VERIFY DT2:*.*/DOS SY:

7.   PDS> COPY INDEXED.DAT/IND/KEY:NUM:2 SEQUEN.DAT
```

CORAL

The CORAL command invokes the CORAL 66 compiler to compile one CORAL 66 source file. Command qualifiers control output file options and subsequent processing.

FORMAT

```
     PDS> CORAL[qualifier(s)]

     FILE?  filespec
```

or

```
     $CORAL[qualifier(s)]  filespec
```

where

filespec	is the specification of a source program file to be compiled. If the filetype is omitted, the system assumes it to be COR. No wild-cards are allowed.
qualifier(s)	are one or more of the following command qualifiers:

Qualifier	Meaning
/LIST[:filespec]	Produce a listing file; name as indicated. If the filetype is omitted from filespec, the system assumes it to be .LST.
/NOLIST	Do not produce a listing file.
/OBJECT[:filespec]	Produce an object file; re-name as specified.
/NOOBJECT	Do not produce an object file.
/SWITCHES:(/sw1.../swn)	Use specified CORAL switch options. For further details, see below.

DEFAULTS:

1. By default, the compiler produces an object file with the name of the source file and with OBJ as the filetype.

2. A listing file is sent to the line printer if /LIST is specified with no filename. /NOLIST is the default qualifier.

CORAL 66 Switches

Switch	Default	Description
/LP:n	/LP:1	Specifies the listing options and 132-column format. n=0 Page headers, switch summary and error diagnostics n=1 As n=0 plus source listing n=2 As n=1 plus CORAL macro expansions n=3 As n=2 plus data and label maps
/TT:n	/TT:1	As for /LP:n but with 80-column format. If neither /LP nor /TT are specified the default is /LP:1. If /TT is specified the default is /TT:1.
/BC	/-BC	Check array, table and switch bounds

/IS:isv	/IS:dis	Specifies instruction set.
		isv=EAE 11/04 with Extended Arithmetic Element
		isv=P45 11/45 instruction set or 11/40 with Extended Instruction set.
		isv=FIS 11/40 with Extended Instruction Set and Floating Instruction Set
		isv=FPP 11/45 with Floating Point Processor
		dis=P45 CORAL 66
		dis=FPP CORAL 66 with FPP support

/OP:n	/OP:1	Specifies optimization of type OP (see PDP-11 CORAL 66 manual) n=0 no optimization of type OP n=1 two passes of OP optimization n=2 iterative passes of OP optimization till no further reduction

/OS:n	/OS:0	Specifies optimization of type OS (see PDP-11 CORAL 66 manual). Current logical registers are retained on meeting:
		n=1 anonymous reference
		n=2 data overlay
		n=4 formal LOCATION parameter

Switch	Default	Description
		With /OS, any combination of the values 1,2,4 can be summed, for example n=3 retain logical registers on meeting anonymous reference or data overlay Default value: n=Ø no optimization of type OS
/SP	/SP	Listing file is queued to the spooler and deleted after print out.
/-SP	/SP	No spooling, listing file is preserved on device indicated by the listing file specification
/TE:n	/TE:Ø	Compile declarations and statements prefixed by 'TEST'm provided n is greater than or equal to m. n is a decimal integer constant and is positive or zero. If /TE is omitted, 'TEST' declarations and statements are ignored.

Switch default summary:

```
(/LP:1/BC/IS:P45/OP:1/OS:Ø/SP/TE:Ø)
```

FURTHER INFORMATION

For further information on the use of the CORAL 66 compiler refer to the following document:

PDP-11 CORAL 66 (with FPP Support) Language Reference Manual and User's Guide

EXAMPLES

1. `PDS> CORAL  NEWFILE`

2. `PDS> CORAL/SW:(/BC/OP:2) FILES.COR`

3. `$CORAL/OBJ:YRFILE.OBJ  MYFILE`

CREATE

The CREATE command creates a file, and (for a sequential file) copies into it source lines following the command in a batch stream or input entered from a terminal. (FORMAT 1)

CREATE can also create an empty file suitable for manipulation by the RMS-11 file services and utilities. Such a file can be filled with the aid of the PDS MERGE command or a user task. (FORMAT 1)

CREATE/DIRECTORY allows the user to create a directory on any volume that is both mounted for him and to which he has write access. (FORMAT 2)

FORMAT 1

```
     PDS> CREATE [/qualifiers-1] newfile [/qualifiers-2]
```

or

```
     PDS> CREATE [/qualifiers-1]

     FILE? newfile [/qualifiers-2]
```

where /qualifiers-1 can be

/DOLLARS	Used only when creating a SEQUENTIAL file in BATCH mode. All batch input up to the next $EOD is used to fill the created file.
/OWN	Causes the destination UFD to be also the owner of the file.

qualifiers-2 can be

```
     /ALLOCATION:n

     /BUCKET_SIZE:n

     /CONTIGUOUS

     /FORMAT:type

     [/INDEXED] /KEY:(parameters)[.../KEY:(parameters)]

     /PROTECTION:(code)

     /RELATIVE

     /SEQUENTIAL
```

where:

/ALLOCATION:n	Forces the file to have n blocks of initial allocation.
/BUCKET_SIZE:m	This qualifier may only be used with INDEXED or RELATIVE and specifies the unit of allocation of this kind of file. m specifies the number of blocks to be allocated to each bucket.

/CONTIGUOUS	Forces the file created to be contiguous.
/FORMAT:type	Specifies the record type of the file. The following types are available.

FIXED:n	fixed length records, n must be specified and is the length of each record in bytes.
VARIABLE[:n]	Variable length records; n may optionally be used to specify the length otherwise the default length of Ø is assumed. If RELATIVE is specified then n must be specified.
CONTROLLED[:n]	Variable length records with a fixed control field. n may be optionally specified to define the length of a record, if not then the default length of Ø is assumed.

/RELATIVE	Specifies relative organisation.
/SEQUENTIAL	Specifies sequential file organisation. If the organisation is not specified, SEQUENTIAL is assumed.
/PROTECTION:(code)	Create the file with the specified protection access code. See Section 6.1.3.
[/INDEXED]/KEY	Create an ISAM file. If INDEXED is specified then NUMBER:1 must appear in one of the key definitions to specify the primary key field.

/KEY is used to specify a key field within the records. /KEY has three mandatory parameters and two optional. Parameters are separated either by spaces, tabs or a comma. The parameter list is written within round brackets.

NUMBER:i	specifies the key field number. i is 1 for the primary key, 2 for first alternate, and 3 for second alternate, etc.
POSITION:j	specifies the starting byte of the key within the record. j=Ø corresponds to the starting byte (byte Ø) of the record.
SIZE:k	determines the length of the key field.

Note that POSITION, SIZE and NUMBER are mandatory.

UPDATE	specifies that the keyfield may change during an update process. NOUPDATE is the converse.
DUPLICATE	specifies that duplicate keyfields may

exist in a record. NODUPLICATE is the converse of DUPLICATE.

Note that if UPDATE is specified then DUPLICATE is implicit.

The following table shows the legal combinatons of DUPLICATE and UPDATE.

Keytype	Combination			
	UPDATE DUPLICATE	UPDATE NODUPLICATE	NOUPDATE DUPLICATE	NOUPDATE NODUPLICATE
Primary	Error	Error	Allowed	Default
Alternate	Default	Error	Allowed	Allowed

If no organisation is specified then SEQUENTIAL is assumed.

If SEQUENTIAL organisation is specified (or defaulted)the user may fill the file with text from his terminal up to CTRL/Z. If in BATCH mode, then /DOLLARS must be used and $EOD will terminate input. No other qualifiers apart from DOLLARS, ALLOCATION:n or OWN may be used. The filespecification must always be fully defined.

The user can specify the BUCKET_SIZE option only for relative and indexed files. A bucket can obtain from 1 to 32 virtual blocks. The default value is one virtual block per bucket. File processing is usually improved if the number is increased.

The relationship between this option and the record size value specified in the record format (/FORMAT) qualifier is important. Since RMS-11 does not allow records to cross bucket boundaries, the user must ensure that the number of virtual blocks per bucket conforms to one of the following formulas:

1. Indexed files with FIXED length records:

   Bnum=(15+(Rlen+7)*Rnum)/512

   where

Bnum	is the number of virtual blocks per bucket, ranging from 1 to 32.
Rlen	is the fixed record length.
Rnum	is the number of records per bucket.

2. Indexed files with VARIABLE length records:

   Bnum=(15+(Rmax+9)*Rnum)/512

   where

Bnum	is the number of virtual blocks per bucket, ranging from 1 to 32.
Rmax	is the maximum size of any record in the file.

Rnum is the number of records per bucket.

3. Relative files with FIXED length records:

   Bnum = ((Rlen+1)*Rnum)/512

   where

Bnum	is the number of virtual blocks per block ranging from 1 to 32
Rlen	is the fixed record length.
Rnum	is the number of records per bucket.

4. Relative files with VARIABLE length records:

   Bnum = ((Rmax+3)*Rnum)/512

   where

Bnum	is the number of virtual blocks per bucket, ranging from 1 to 32.
Rmax	is the maximum size of any record in the file
Rnum	is the number of records per bucket.

5. Relative files with controlled format records:

   Bnum = (Rmax+Fsiz+3)*Rnum/512

   where

Bnum	is the number of virtual blocks per bucket, ranging from 1 to 32
Rmax	is the maximum size of the data portion of any CONTROLLED record in the file.
Fsiz	is the size of the fixed control area portion of the CONTROLLED records.
Rnum	is the number of records per bucket. CONTROLLED records in a relative file bucket always occupy Rmax+Fsiz+3 bytes.

In all cases, if Bnum is not an integer, it must be rounded up to the next integer value.

DESCRIPTION

Batch Mode

In batch mode the text to be placed in the new file follows the command. Any $ command terminates the file unless the CREATE command string includes the qualifier /DOLLARS, which specifies that only the command $EOD can terminate the file.

Interactive Mode

The CREATE command reads the input to the new file from the user's terminal. Pressing CTRL/Z terminates the file.

For SEQUENTIAL files, the CREATE command has the same function as a COPY command that specifies TI: as the device in the input file specification.

EXAMPLES

1.
```
PDS> CREATE

FILE?  MYDATA.DAT;5

READY FOR INPUT

ABCD

EFGH

CTRL/Z

PDS> CREATE ANOTHER.DAT/PROTECTION:(OW:RW)

READY FOR INPUT

CTRL/Z

PDS>
```

2.
```
$CREATE/DOLLARS   DEBUG.MAC

.
.
.

$EOD
```

3.
```
PDS> CREATE JOHN.DOE/KEY:(NUMBER:1,SIZE:10,POSITION:0)
```

Creates JOHN.DOE as an Indexed Sequential (ISAM) file having variable lenth records with one key of reference. The key is 10 (decimal) bytes long and appears in the first byte (byte 0) of each record.

FORMAT 2

```
     PDS> CREATE/DIRECTORY/ALLOCATION:n

     DEVICE? dev:uic[/PROTECTION:(protection)]
```

where

n	is the number of files for which room is initially allowed in the directory. The file system will extend the directory file as needed, if this value is subsequently exceeded.
dev	is the device on which the directory is to be created.
uic	is the UIC to be given a UFD on the device.
protection	is the file protection to be placed on the directory. Protection is specified in the form:

(SYSTEM:RWED,OWNER:RWED,GROUP:RWED,WORLD:RWED)

Access protection is allocated for four groups:

SYSTEM	- tasks running under a UIC with group number 10 octal or less.
OWNER	- tasks running under the same UIC as in the owner field of the file.
GROUP	- tasks running under UICs with the same group number as the owner.
WORLD	- all tasks.

The access protection that can be specified for each of the four groups is:

R - Read access

W - Write access

E - Extend access

D - Delete access

Specifying an access for a group allocates the access rights to the group. If a group is omitted from the protection, the volume default position for that group is allocated.

EXAMPLES

1. `PDS> CREATE/DIRECTORY DK1:[11,17]`

This example will create a directory under [11,17] on DK1:.

2. `PDS> CREATE/DIRECTORY/ALLOCATION:6 LB:[14,7]`

This example will create a directory under [14,7] on LB: with space initially allowed for 6 files.

3. `PDS> CREATE/DIR DPØ:[2ØØ,2ØØ]/PRO:(SYSTEM:RWED,OWNER:RWED,WORLD:)`

This example creates a directory of [2ØØ,2ØØ] on disk DPØ. The access protection is RWED for System, RWED for Owner ([2ØØ,2ØØ]), the volume default Group protection for Group and no access for World.

4. `PDS> CREATE/DIR [123,22]`

This example will create a directory of [123,22] on the users current default device.

DEALLOCATE

The DEALLOCATE command deallocates a specified device.

FORMAT

```
    PDS> DEALLOCATE

    RESOURCE?  DEVICE

    DEVICE?  device-name
```

or

```
    $DEALLOCATE DEVICE device-name
```

where

device-name	is the device specification or the logical name of the device to be deallocated.

DESCRIPTION

Normally the system automatically deallocates a device when the user dismounts the volume on it or deassigns it from a logical unit number. However, when the user has issued the ALLOCATE command to obtain access to a non-mountable device that has not been assigned to a logical unit, the DEALLOCATE command must be used to release it. It may also be used after a DEASSIGN/KEEP or DISMOUNT/KEEP command.

EXAMPLES

1. `PDS> DEALLOCATE DEVICE DK1:`

2. `$DEALLOCATE DEVICE DD0:`

DEASSIGN

The DEASSIGN command dissociates a device from a logical unit.

FORMAT

```
    PDS> DEASSIGN[/KEEP]

    LUN? lun
```

or

```
    $DEASSIGN[/KEEP] lun
```

where

/KEEP	inhibits any deallocation or dismounting of the associated device.
lun	is the logical unit number to be deassigned.

DESCRIPTION

If the specified logical unit number is the last to which a device is assigned, the device is dismounted or deallocated unless the user specifies the command qualifier /KEEP.

The command applies only to assignments made for timesharing user tasks (i.e. not ASSIGN/REAL). See the ASSIGN command for deassignment of REALTIME assignments.

EXAMPLES

1.
   ```
   PDS> DEASSIGN
   LUN? 7
   ```

2.
   ```
   $DEASSIGN/KEEP 3
   ```

DELETE

The DELETE command deletes one or more specified files.

FORMAT

```
     PDS> DELETE[/KEEP:n]

     FILE?  filespec-l[file-qualifier][,...filespec-n]
```

or

```
     $DELETE[/KEEP:n] filespec-l[file-qualifier][,...filespec-n]
```

where

KEEP[:n]	prevents the latest n versions of a specified file from being deleted. It can only be used when the version field in a file specification is omitted or wild. If n is omitted, it is assumed to be 1.

NOTE

If a DELETE/KEEP is attempted on files that are protected, because of the directory structure, the system will attempt many times to delete the file. The user should press CTRL/O and wait or abort the operation.

filespec	is the file specification of a file to be deleted. Wild-cards are allowed. The version field MUST be specified unless /KEEP is used or the file is foreign.
file-qualifier	modifies the specification of a foreign file in DIGITAL'S DOS or RT-11 format. The qualifiers are: /DOS /RT11

DESCRIPTION

The user cannot recover a deleted file.

In order to delete a file in DOS or RT11 format, the user must modify the file specification with the /DOS or /RT11 file qualifier.

EXAMPLES

```
1.    PDS> DELETE (A.EXT;1, B.EXT;1, DKØ:C.*;*)

2.    PDS> DELETE/KEEP:1
      FILE?  DK1:[2ØØ,2ØØ]*.XYZ

3.    $DELETE/KEEP DKØ:[2ØØ,2ØØ]*.LIS

4.    PDS> DELETE DTØ:TEST.MAC/DOS
```

DIRECTORY

The DIRECTORY command lists details about a file or a group of files at a specified output device or to a specified file. Command qualifiers allow the user to request greater or less detail.

FORMAT

```
     PDS> DIRECTORY[qualifier(s)]

     FILE? filespec-1[file-qualifier][,..filespec-n]]
```

or

```
     $DIRECTORY filespec-1[file-qualifier][,...filespec-n]
```

where

filespec	is a file specification that indicates the directory entries to be listed. Wild-cards are allowed. If no files are specified, the system lists information about all the files in the user's default directory.
qualifier(s)	are one or more of the following:

Qualifier	Meaning
/OUTPUT:outfile	List information in the specified output file.
/SUMMARY	Specify that only a summary line of the following format is required:

TOTAL OF nnnn./mmmm.BLOCKS IN xxxx.FILES

where

nnnn = blocks used
mmmm = blocks allocated
xxxx = number of files

/BRIEF	List only the name, type and version of the file(s).
/FREE	Show free space available on the user's default device.

NOTE

If the volume concerned was initialized under RSX-11D V4A when the /FREE qualifier is used, access will not be permitted.

/FULL	List all the following file details: 1. Name, filetype and version 2. File identification number in the format:(file number, file sequence number) 3. Number of blocks used or allocated. 4. File code null = non-contiguous C = contiguous L = locked 5. Creation date and time 6. Owner UIC and file protection in the format: [group, owner] [system, owner, group, world] 7. Date and time of the last update and the number of revisions.
/PRINT	Output the directory listing to the line printer.

file-qualifier   modifies a foreign file in DIGITAL'S DOS or RT-11 format. The qualifiers are:

/DOS
/RT11

DESCRIPTION

By default, the DIRECTORY command lists at the user's terminal (interactive mode) or in the user's output stream (batch mode) the name, filetype, version, size, file code, and date and time of creation of all the files in the directory which is the user's current default.

The user may not examine the files in a directory for which he does not have read access.

To interrogate the directories of DOS or RT-11 volumes, the user must modify the file specification with the /DOS or /RT-11 file qualifier.

The directory listing of a DOS or RT-11 file corresponds to the directory format of the foreign volume. The qualifiers /BRIEF and /FULL are not valid when requesting foreign directory information.

When a directory listing of a Files-11 (ANSI) magnetic tape is produced, the creation time for all files appears as 00:00. This is because there is no place for the creation time of a file in the ANSI file header label.

EXAMPLES

```
1.    PDS> DIRECTORY <ALT>

      FILE? MATRIX.DAT/DOS

2.    PDS> DIR/FULL/OUTPUT:DKØ:DIR.DAT <ALT>

      FILE?  DK1:[2ØØ,2ØØ]*.LST

3.    $DIR/BR FRED.*

4.    $DIRECTORY DK1:*.*/RT11
```

DISABLE

The DISABLE command allows the user with PDS Command Privilege PR.RTC to inhibit task execution of an installed task without actually removing the task from the system. Disabled tasks cannot be initiated until they are enabled through the ENABLE command, (see the IAS Executive Reference Manual, Volume II, Chapter 6) or the ENABLE directive.

FORMAT

```
    PDS> DISABLE

    TASK?  taskname
```

where

taskname	is the installed name of the task being disabled.

EXAMPLES

```
    PDS> DISABLE XKE2Ø

    PDS> DISABLE MYJOB2
```

DISMOUNT

The DISMOUNT command causes the volume on the specified device to be dismounted.

FORMAT

```
     PDS> DISMOUNT[/qualifier]

     DEVICE? device-name

     [VOLUME-ID? volume-identification]
```

or

```
     $DISMOUNT device-name volume-identification
```

where

qualifier is one of the following:

/KEEP	instructs the system not to deallocate the device
/GLOBAL	instructs the system to dismount the globally mounted device
/REALTIME	instructs the system to dismount the device that was mounted for exclusive access by real time tasks
device-name	is the physical or logical name of the device to be dismounted.
volume-identification	is an optional parameter that specifies the identification of the volume to be dismounted (see the MOUNT command).

DESCRIPTION

If the user does not specify /KEEP, the system dismounts the volume on the device, deallocates the device, and deassigns it from any logical unit number.

If the qualifiers /GLOBAL or /REALTIME are omitted, the default action of the DISMOUNT command is to dismount the volume for the timesharing user who issues the command. In this case the system:

1. Dismounts the volume from the device for the user
2. Deallocates the device if it was previously allocated (unless /KEEP is used).
3. Deassigns the device from any logical unit number(s) that the user has assigned.

If the user is the last of several who are sharing the volume and the volume was globally mounted then a full/final dismount occurs. The operator is then requested to unload the volume from the device. If the volume is mounted globally then the full/final dismount will not occur until an explicit DISMOUNT/GLOBAL is issued.

Only if the device was allocated to a user (by means of the ALLOCATE command) then the command qualifier /KEEP will prevent the system from deallocating the device.

EXAMPLE

1.
```
PDS> DISMOUNT

DEVICE? MYØ:
```

2.
```
$DISMOUNT/KEEP TU1:  ACCTS
```

DUMP

The DUMP command produces a printed listing of the contents of a file. DUMP ignores any print formatting characters that may appear in the records. The listing is printed at the user's terminal by default, but the user may specify a different output device.

FORMAT

```
     PDS> DUMP[qualifier(s)]

     FILE?  filespec
```

or

```
     $DUMP[qualifier(s)] filespec
```

where

filespec	is the specification of the file or device to be dumped.
qualifier(s)	are one or more of the following command qualifiers:

Qualifier	Meaning
/OUTPUT:filespec	Output the listing to the specified file or device.
/ASCII	The /ASCII switch specifies that the data should be listed in ASCII mode. The control characters (Ø - 37) are printed as ^ followed by the alphabetic character corresponding to the character code +1ØØ(octal). For example, bell (code 7) is printed as ^G (code 1Ø7). Lower case characters (14Ø - 177) are printed as % followed by the corresponding upper case character (character code minus 4Ø).
/BLOCKS:(m-n)	Specifies the first (m) through the last (n) logical or virtual blocks to be listed, where m and n are octal numbers. If either m or n is greater than 16 bits (that is, greater than 177777) then the user must specify it as two numbers as follows: (a,b) where a is the first 16 bits and b is the second 16 bits. If the /BLOCKS:(m-n) switch is specified as /BLOCK:Ø in file mode, no physical blocks will be listed. This is useful when the user wishes to list only the header portion of the file. (See the /HEADER switch below). This qualifier is necessary in device mode; it specifies the range of physical blocks to be listed.

Qualifier	Meaning
/BYTE	The /BYTE qualifier specifies that the data should be listed in byte octal format.
/HEADER	If specified, /HEADER causes the file header as well as the specified portion of the file to be listed. If just the header portion of the file is required, the user can specify /HEADER/BLOCKS:Ø.

/START	This qualifier gives the user only the starting block number of the file and an indication of whether or not it is contiguous.
	
	Example:
	
	`DUMP/START DKØ:RICKSFILE.DAT;3`
	`STARTING BLOCK NUMBER = Ø.135163 C`
	
	File RICKSFILE.DAT, version 3 is a contiguous file starting at block no. Ø,135163. (See /BLOCKS:(m-n) for a description of block and number format.)

/NUMBER[:n]	This qualifier allows control of line numbers. Line numbers are normally reset to zero whenever a block boundary is crossed. The /NUMBER[:n] qualifier allows lines to be numbered sequentially for the full extent of the file; i.e., the line numbers are not reset when block boundaries are crossed. The optional value (:n) allows the user to specify the value of the first line number. The default is Ø.
/PRINT	Output the listing to the default printer.

DESCRIPTION

The DUMP command operates in either one of two modes:

1. File Mode

   In file mode, the user specifies a file; all, or a specified range (see(/BLOCKS:(m-n)) of blocks of the named file are listed. The blocks are numbered from 1 through n, where the first block is 1 and the last block in the file is numbered n. The input volume must be mounted and it must contain named files. Wild filenames are not permitted.

2. Device Mode

   In device mode, the user specifies a device; then a specified range (/BLOCKS:(m-n)) of physical blocks to be listed.

   a. The /BLOCKS:(m-n) qualifier is required.

b. Physical blocks refer to the actual 512-byte blocks on disk and DECtape, and physical records on magtape and cassette. The DUMP command handles physical records up to 2Ø48 bytes in length.

c. Physical blocks are numbered from Ø to n, where n is the last logical block on the device.

d. The volume to be listed must be mounted as FOREIGN.

EXAMPLES

```
1.   PDS> DUMP MYFILE.DAT

2.   PDS> DUMP/ASCII

     FILE?  MYDATA.DAT

3.   $DUMP A.MAC;4

4.   PDS> DUMP/BLOCK:(5-14) DKØ:
```

EDIT

The EDIT command invokes one of the following IAS text editors:

1. The Line Text Editor (EDI), an editor primarily for interactive use

2. The Source Language Input Program and Editor (SLIPER), a batch-oriented editor.

Chapter 7 describes how to use both editors. The <u>IAS Editing Reference Manual</u> specifies both in full.

FORMAT

```
    PDS> EDIT[/editor][qualifier(s)]

    FILE?  filespec
```

or

```
    $EDIT[/editor][qualifier(s)] filespec
```

where

/editor — is either:

/EDI which invokes the Line Text Editor, or

/SLIPER which invokes the batch editor SLIPER

The default is /EDI

qualifier(s) — are one or more command qualifiers that are only valid if /SLIPER has been specified. The qualifiers are described in detail in Chapter 7, Section 7.2.1. They are:

Qualifier	Default
/OUTPUT[:filespec] /NOOUTPUT	/OUTPUT
/LIST[:filespec] /NOLIST	/LIST (batch mode) /NOLIST (interactive mode)
/AUDIT[:(params)] POSITION:m SIZE:n /NOAUDIT	/AUDIT
/BLANK /NOBLANK	/BLANK
/DOUBLE /NODOUBLE	/NODOUBLE

filespec — is the specification of an existing file to be edited or a new file to be created. A filetype must be included.

The Line Text Editor (EDI)

The Line Text Editor is described in Chapter 7. A complete specification is contained in the IAS Editing Utilities Reference Manual. This section lists all the editor commands that can be issued once the user has invoked the Line Text Editor.

ADD A[DD] string	Add the text specified by "string" to the end of the current line.
ADD AND PRINT AP string	Same as ADD, except the new current line is printed out.
BEGIN B[EGIN]	Sets the current line pointer to the top of the block buffer or input file.
BLOCK ON or OFF BL[OCK][ON] or [OFF]	Switch editing modes.
BOTTOM BO[TTOM]	Sets the current line pointer to the bottom of block buffer or input file.
CHANGE [n]C[HANGE] /string-1/string-2	Search for string-1 and replace it with the text specified in string-2. n allows the user to repeat the command, thus allowing string-2 to be substituted for string-1 n times within the current line.
CLOSE CL[OSE]	Transfer the remaining lines in the block buffer and the input file into the output file, then close both the input file and the output file.
CLOSES CLOSES	Close secondary input file, and begin selecting lines from the input file.
CLOSE AND DELETE CDL	Same as the CLOSE command except that the input file is deleted.

CONCATENATION CHARACTER CC character	Change command concatenation character to the actual specified character (default is &).
DELETE D[ELETE] [n] or [-n]	Delete the current and next n-1 lines, if n is positive; delete n lines preceding the current line, but not the current line, if n is negative.
DELETE AND PRINT DP [n] or [-n]	Same as DELETE except that the new current line is printed out.
END E[ND]	Same as the BOTTOM command.
ERASE ERASE [n]	Erase the entire block buffer, the current line, and the next n blocks.
EXIT EX[IT]	Same as CLOSE command.
EXIT AND DELETE EDX	Exit from the editing session, close the output file, delete the input file.
FORM FEED FF	Insert form feed into block buffer.
FILE FI[LE] filespec	Transfer lines from the input file to the file specified by filespec.
FIND [n]F[IND] string	Find the line starting with "string" or, if n is specified the nth line starting with "string".
INSERT I[NSERT] [string]	Insert "string" immediately following the current line. If "string" is null, EDI enters Input Mode.

KILL KILL	Terminate this editing session; close input and output files; delete the output file.
LINE CHANGE [n]LC /string-1/string-2	Same as CHANGE except that all occurrences of string-1 in the current line are changed to string-2.
LIST ON TERMINAL LI[ST]	Print on user terminal all lines in block buffer or all remaining lines in input file, starting with current line.
LOCATE [n]L[OCATE] string	Search the block buffer for "string" or, if n is specified the nth occurrence of "string".
MACRO MA[CRO] x definition	Define macro x to be "definition".
MACRO CALL MC[ALL]	Retrieve macros from the latest version of file MCALL.EML.
MACRO EXECUTE [n]Mx[a]	Execute Macro x for n executions passing it the numeric argument a.
MACRO IMMEDIATE [n]<definition>	Immediate Macro - this allows the user to define and execute a macro in one step.
NEXT N[EXT] [n] or [-n]	Establish a new current line + or - n lines from the current line.
NEXT PRINT NP [n] or [-n]	Next Print; same as Next command, but the new current line is printed out.
OLD PAGE OL[DPAGE] n	Back up to page n.

OPENS OPENS filespec	Open secondary input file.
OUTPUT ON or OFF OU[TPUT] [ON] or [OFF]	Turn output on or off.
OVERLAY O[VERLAY] [n]	Delete the current line and the next n-1 lines, and enter Input Mode.
PAGE PAG[E] [n]	Enter block edit mode, if not already in block edit mode, and read page n into the block buffer.
PAGE FIND [n]PF[IND] string	Identical to FIND command except that it searches successive pages until the nth occurrence of "string" is found.
PAGE LOCATE [n]PL[OCATE]string	Same as LOCATE command, except that successive pages are searched for the value specified by "string".
PASTE PA[STE] /string-1/string-2	The same as the LINE CHANGE command except that all lines in the remainder of the input file or block buffer are searched for string-1. Wherever found, string-1 is replaced with string-2.
PRINT P[RINT] [n]	Print out the next line, and the next n-1 lines, on the terminal.
READ REA[D] [n]	Read the next n pages into the block buffer.
RENEW REN[EW] [n]	Write the current buffer, and read in the next page.

RETYPE R[ETYPE] [string]	Replace the current line with the text of "string". If "string" is null; the line is deleted.
SAVE SA[VE] [n] [filespec]	Save the current line, and the next n-1 lines, in the file specified by filespec.
SEARCH & CHANGE SC /string-1/string-2	Search for string-1, in the block buffer or input file starting with the line following the current line. When string-1 is found, replace all occurrences in line with string-2.
SELECT PRIMARY SP	Select primary input file.
SELECT SECONDARY SS	Select secondary input file.
SIZE SIZE n	Specify maximum number of lines to be read into the block buffer on a single READ.
TAB ON or OFF TA[B] [ON] or [OFF]	Turn automatic tabbing on or off.
TOP T[OP]	Same as BEGIN command.
TOP OF FILE TOF	Returns to the top of the input file, in block edit mode, and saves all pages previously edited.
TYPE TY[PE] [n]	Same as PRINT command except that the current line pointer does not change.
UNSAVE UNS[AVE] [filespec]	Retrieve the lines which were previously saved on filespec and insert them

	immediately following the current line.
UPPER CASE ON or OFF UC ON	Convert all input characters to upper case (default state).
UC OFF	Accept all input without case conversion.
VERIFY ON or OFF V[ERIFY] [ON] or [OFF]	Allows user to select whether or not locative and change commands are to be verified.
WRITE W[RITE]	Write the current block to the output file, and erase the contents of the buffer.

The Source Language Input Program and Editor (SLIPER)

The SLIPER edit control characters are as follows:

Character	Function
/(slash)	The slash is placed in the first position of a line to indicate that the editing of a file is completed.
@(at)	The @ character is put in the first location of a line to indicate that SLIPER is to seek input from an indirect file. The user must specify the indirect file immediately after the @ sign; for example: @DK2:DKSFIL.CMD instructs SLIPER to read input from the file DKSFIL.CMD on physical device unit DK2:. Indirect files are more fully described in Section 7.2.4.
<(less than)	The < character is used when entering a line that begins with one of the special edit control characters. It causes the line to be shifted one character to the left, with the result that < is deleted, and the desired control character becomes the first character on the line.

For an insertion:

```
-location1[,/audit-trail][;]
```

Insert text following the line in the input file given by location1.

For a deletion:

```
-location1,location2[,/audit-trail][;]
```

Delete line(s) given by location1 through location2.

where:

location1 and location2 are

n	n is a line number (decimal)
or	
/string/[+n]	string is an ASCII string and may occur anywhere in the line to be located. Within string, three periods ... can be used to represent omitted characters. +n, if used, advances the location a further n (decimal) lines.
or	
.[+n]	current position [advanced in lines].
audit-trail	is an ASCII string to be appended to each new line of text if /AUDIT is in force. Default (if /AUDIT is in force) is the immediately previous setting of audit-trail. Initial setting: ;**NEW**
;	remainder of line following ; is a comment.

ENABLE

The ENABLE command reverses the effects of the DISABLE command.

FORMAT

```
PDS> ENABLE
TASK?  taskname
```

where

taskname	is the installed name of the task being enabled.

EXAMPLES

```
PDS> ENABLE XKE2Ø

PDS> ENABLE MYJOB2
```

$EOD

The $EOD (End of Data) command terminates a data stream or the input to a file created by a $CREATE/DOLLARS command.

FORMAT

```
    $EOD
```

The command has no parameters.

EXAMPLE

```
    $CREATE/DOLLARS PAYROLL.DAT
    ; PAYROLL UPDATE FOR 27-JAN
    DOE JOHN
    $476.32    $46.12    17    P
    BLOGGS FRED
    $316.41    $96.24    23    R
    $EOD
```

This example uses $EOD to terminate a file of batch commands (an indirect file). The /DOLLARS qualifier instructs the system to accept the following lines of text as input to the file rather than batch commands to be processed.

$EOJ

The $EOJ (End of Job) command terminates a batch job, dismounting and releasing any claimed devices.

FORMAT

```
$EOJ
```

The command has no parameters.

DESCRIPTION

THE $EOJ command must be the last command in a batch job command stream.

EXAMPLE

```
$JOB WILSON TESTRUN 2

$MOUNT DK:  TEST DDØ:

$ASSIGN DDØ: 7

$RUN TEST

$DISMOUNT DDØ:

$EOJ
```

FIX

The FIX command allows the user to fix a task in its installed partition. The main benefit of fixing tasks is that there is no delay while the task is loaded for the first time. Also, memory fragmentation can be prevented by fixing tasks in a system-controlled partition. The user can only "fix" a task if the task was built as a fixable task (see the IAS Task Builder Reference Manual).

FORMAT

```
PDS> FIX
TASK?  taskname
[TERMINAL?  terminal]
```

where

taskname	is the installed name of the task to be fixed in memory.
terminal	is the terminal for which the task is to be fixed. It is possible to fix the same task for more than one terminal. Also, a task can be fixed for one terminal and not fixed for another.

EXAMPLES

```
PDS> FIX MYTSK

PDS> FIX MART3 TT4
```

FORTRAN

The FORTRAN command invokes a FORTRAN compiler to compile one FORTRAN-IV or FORTRAN-IV PLUS source file. Command qualifiers control output file options and subsequent processing.

FORMAT

```
     PDS> FORTRAN[qualifier(s)]

     FILE?  filespec
```

or

```
     $FORTRAN[qualifier(s)]  filespec
```

where

filespec	is the specification of a source program file to be compiled. If the filetype is omitted, the system assumes it to be FTN. No wild-cards allowed.
qualifier(s)	are one or more of the following command qualifiers:

Qualifier	Meaning
/FOR	Invoke the FORTRAN-IV compiler. Applicable to systems that have both FORTRAN IV and FORTRAN IV PLUS compilers. If omitted, the system invokes its default compiler.
/F4P	Invoke the FORTRAN IV-PLUS compiler. Applicable to systems that have both FORTRAN IV and FORTRAN IV PLUS compilers. If omitted, the system invokes its default compiler.
/LIST[:filespec]	Produce a listing file; name as indicated. If the filetype is omitted from filespec, the system assumes it to be .LST.
/NOLIST	Do not produce a listing file.
/OBJECT[:filespec]	Produce an object file; re-name as specified.
/NOOBJECT	Do not produce an object file.
/SWITCHES:(/sw1.../swn)	Use specified FORTRAN IV or FORTRAN IV-PLUS switch options. For further details, see below.

DEFAULTS:

1. By default, the compiler produces an object file with the name of the source file and with OBJ as the filetype.

2. A listing file is sent to the line printer if /LIST is specified with no filename. /NOLIST is the default qualifier.

FORTRAN-IV Switches

Switch	Default	Description
/LI:n	/LI:3	Specifies the listing options. The argument n is encoded as follows: /LI:Ø or /NOLI — list diagnostics only /LI:1 or /LI:SRC — list source program and diagnostics only /LI:2 or /LI:MAP — list storage map and diagnostics only /LI:4 or /LI:COD — list generated code and diagnostics only Any combination of the above list options may be specified by summing the numeric argument values for the desired list options. For example: /LI:7 or /LI:ALL requests a source listing, a storage map listing, and a generated code listing. If this switch is omitted the default list option is /LI:3, source and storage map.
/DE	/NODE	Compile lines are with a D in column one. These lines treated as comment lines by default.
/EX	/NOEX	Read a full 8Ø columns of each record in the source file. Only the first 72 columns are read by default.
/ID	/NOID	Print FORTRAN identification and version number. The default (/NOID) causes the identification and version number not to be printed.
/OP	/OP	Enable the Common Subexpression Optimizer (CSE). In general the CSE optimizer will make the program run faster. However, the size of the program may be different than with no optimization.
/SN	/SN	Include internal sequence numbers (ISN). The option reduces storage requirements for generated code and slightly increases execution speed but disables line number information during Traceback.

Switch	Default	Description
/I4	/NOI4	Two word default allocation for integer variables. Normally, single storage words will be the default allocation for integer variables not given an explicit length specification (i.e., Integer*2 or integer*4). Only one word is used for computation.
/VA	/VA	Enable vectoring of arrays (see Section 2.5 of the FORTRAN IV User's Guide).
/WR	/WR	Enable compiler warning diagnostics.

Switch default summary:

(/LI:3/NODE/NOEX/NOID/OP/SN/NOI4/VA/WR)

FORTRAN-IV Plus Switches

Switch	Default	Description
/CK	/NOCK	Code is generated to check that all array references are within the array bounds specified by the program. Individual subscripts are not checked against dimensional specifications.
/CO:n	/CO:5	A maximum of n continuation lines is permitted in the program, where n is from Ø through 99. The default value is n=5. Note that n may be expressed either in octal or decimal radix. If a decimal point follows the number, it is interpreted in decimal radix; otherwise, it is interpreted in octal radix.
/DE	/NODE	Compile lines with a D in column one. These lines are treated as comment lines by the default /NODE (see the FORTRAN Language Manual).
/ID	/NOID	Print FORTRAN IV-PLUS identification and version number.
/I4	/NOI4	Allocates two words for default length of Integer and Logical variables. Normally, single storage words will be the default allocation for all Integer or Logical variables not given an explicit length definition (i.e., INTEGER*2, LOGICAL*4). See Section 3.3 of the FORTRAN IV-PLUS User's Guide.
/LI:n	/LI:2	Specifies listing options; n is from Ø through 3. The argument is coded as follows: n=Ø minimal listing file: diagnostic messages and program section summary only

n=1	source listing and program section summary
n=2	(default) source listing, program section summary and symbol table
n=3	source listing, assembly code, program section summary, and symbol table

/TR:XXX /TR:BLOCKS The /TR switch controls the amount of extra code included in the compiled output for use by the OTS during error traceback. This code is used in producing diagnostic information and in identifying which statement in the FORTRAN source program caused an error condition to be detected at execution. /TR:XXX can have the following forms:

/TR	Same as TR:ALL
/TR:ALL	Error traceback information is compiled for all source statements, and function and subroutine entries.
/TR:LINES	Same as ALL option.
/TR:BLOCKS	Traceback information is compiled for subroutine and function entries and for selected source statements. The source statements selected by the compiler are initial statements in sequences commonly called basic blocks. The compiler treats such a sequence of statements as a unit for performing certain types of optimization. Basic blocks generally begin at each labelled statement, each DO statement, and so on.
/TR:NAMES	Traceback information is compiled only for subroutine and function entries.
/TR:NONE	No traceback information is produced.
/NOTR	Same as NONE.

The switch setting /TR is generally advisable during program development and testing. The default setting /TR:BLOCKS is generally advisable for most programs in regular use. The setting /NOTR may be used for obtaining fast execution and smallest code, but it provides no information to the OTS for diagnostic message traceback.

Compiler switch default summary:

(/NOCK/CO:5/NODE/ID/NOI4/LI:1/TR:BLOCKS)

FURTHER INFORMATION

For further information on the use of the FORTRAN sytems, refer to the following documents:

<u>PDP-11 FORTRAN Language Reference Manual</u>

<u>IAS/RSX-11 FORTRAN-IV User's Guide</u>

<u>FORTRAN IV-Plus User's Guide</u>

EXAMPLES

1. <u>PDS></u> FORTRAN NEWFILE

2. <u>PDS></u> FORTRAN/SW:(/CK/CO:7) FILES.FTN

3. $FORTRAN/OBJ:YRFILE.OBJ MYFILE

GOTO

The GOTO command is used only in an indirect command file or a batch command file. GOTO transfers control to the next following occurence of a command line prefixed by a specified label.

FORMAT

```
     [$]GOTO label
```

where

label	is an alphanumeric string and must also appear, together with a colon, in front of a later command in the file.

DESCRIPTION

GOTO can be used by itself or as an action in an ON command. When control is transferred, the system ignores all intervening commands, in particular any intervening ON commands. If no matching label is found, no further processing takes place within the command file or batch job. GOTO cannot transfer control to an earlier labelled command.

EXAMPLE

```
     $JOB SYSTEM
     $ON ERROR GOTO L1Ø
     $MACRO MYPROG
     $LINK MYPROG
     $RUN MYPROG
     $GOTO L2Ø
     $L1Ø:  RUN OLDPROG
     $L2Ø:  RUN TEST
     $EOJ
```

HELP

The HELP command displays information at an interactive terminal to assist the user in issuing PDS commands.

FORMAT

```
     PDS> HELP
```

DESCRIPTION

The precise information displayed depends on the user's current state.

Before the user is logged in, typing HELP causes a display of information on how to log in.

When the user is logged in, the HELP command provides help at a number of levels:

1. A HELP command with no parameters gives a listing on the terminal of all PDS commands.

2. To obtain information on the format of a specific command, supply the required command name as a parameter to the HELP command; e.g.,

   ```
   PDS> HELP LIBRARIAN
   ```

   The format of the command (in this example the LIBRARIAN command) and a list of the relevant qualifiers and parameters will be listed.

   Further information about qualifiers and parameters for the command can be obtained by supplying the qualifier or parameter name as an additional parameter to the HELP command; e.g.,

   ```
   PDS> HELP LIBR EXTRACT
   ```

   This command will provide full details of the EXTRACT feature of the IAS Librarian.

   Only those qualifiers and parameters which HELP flags by two asterisks (**) can be supplied as the additional parameter.

   Again, when the user has suspended a task, HELP can display the options available.

INITIALIZE

The INITIALIZE command is used to initialize a foreign (DOS and RT11) volume. The device must first be allocated to the user mounted /FOREIGN, then INITIALIZE can be used to zero the volume.

FORMAT 1

```
PDS> INITIALIZE/DOS device-spec
```

where

device-spec	is the device on which the DOS volume is to be initialized.

FORMAT 2

```
PDS> INITIALIZE/RT11[:n][/NUMBER:m] device-spec
```

where

:n — is the number of extra words required per directory entry. A directory segment consists of 2 disk blocks or 512 words. The directory header uses 5 words, leaving 507 words for directory entries. Normally, each directory entry is seven words long and two directory segments are allocated to the file system. Therefore, the number of entries in each segment when no extra words are specified is determined as follows:

```
Directory entries = (507 divided by 7)-2
                  = 72-2=70 entries
```

When extra words are specified for directory entries, the number of directory entries is determined as follows:

```
Directory entries = (507 divided by (N+7))-2
```

/NUMBER:m — is used to specify the number of directory segments to allocate to the RT11 volume. The default is four directory segments.

device-spec — is the device on which the RT11 volume is to be initialized.

EXAMPLES

1.
```
PDS> MOUNT/FOREIGN DKØ:  MYDOSDISK

PDS> INIT/DOS DKØ:
```

2.
```
PDS> MOUN/FOR DT RT11SOURCE RT

PDS> INIT/RT11:6/NUMBER:3 RT:
```

3.
```
PDS> MOUNT/FOR/NOOPER DTØ:  DOSDECTAPE MY

PDS> INIT/DOS MYØ:
```

INSTALL

The INSTALL command causes the system to find and note the physical position of a task or Shareable Global Area (SGA) held on disk. This allows fast loading into memory. A task cannot be run in realtime processing (see PDS> RUN) unless it has been installed. Further, a task cannot be installed until all the SGAs which it uses have been installed. The effect of INSTALL is reversed by the REMOVE command.

At installation the user can take the opportunity to override certain task attributes set at link time and to specify non-owner access rights to an SGA. These changes affect only the installed version, not the task or SGA task image file.

FORMAT

The following command is used to install a task or an SGA:

```
PDS> INSTALL[/qualifier1[:newname]][qualifiers2]

FILE? filespec
```

where

/qualifier1	is one of /TASK /COMMON to install a common SGA /LIBRARY to install a library SGA
newname	is a 1- to 6- character name and optionally overrides the name of the task or SGA set at link time
qualifiers2	is one or more of the following qualifier options:

/PARTITION:name
: to install a task or SGA in the specified partition.

/POOL:number
: to set the pool limit of the task to be installed. The pool limit value can range from 0 to 255 decimal and represents the maximum number of 8-word nodes that the task is allowed to use at one time.

/PRIORITY:number
: to set the execution priority to be assigned to the task. Priority ranges from 1 to 250.

/UIC:uic
: to change the task's UIC or the owning UIC of an SGA.

Defaults for the above are the values as set at link time. See the IAS Task Builder Manual, Chapter 3.

/INCREASE:tasksize-increment
to override the EXTTSK option specified in the LINK command. This qualifier specifies the decimal number of words by which the upper read/write area of the task being installed is to be extended. The value specified will be rounded up to the next 32-word boundary.

/ACCESS:ACCESS
is the non-owner access permitted to the SGA being installed:

RO - Read-Only
RW - Read/write
NA - No access by non-owners (default)

The owner always has RW access.

filespec   is the file specification of the task being installed. If the file type is ommitted, a default of TSK is assummed.

EXAMPLES

1. PDS> INSTALL [11,1]PIP

2. PDS> INSTALL/TASK:JK3Ø4/PRIORITY:2ØØ DK1:COMMS.TSK;3

   Install the task image held in file COMMS.TSK;3 on DK1. Give it the installed name JK3Ø4 and priority 2ØØ.

3. PDS> INSTALL/COMMON:COMLOL/ACCESS:RO JK61.TSK;4

   Install the SGA held in file JK61.TSK;4 on the users default device. give it the installed name COMLOL and give Read-Only access to non-owners of the SGA.

4. PDS> INSTALL/LIBRARY:SYSRON/ACCESS:RW DK2:JOHN4.MAC

   Here a library SGA is (unusually) given Read-Write access for non-owners.

5. PDS> INSTALL/TASK:$$$LOL/INCREASE:2Ø48 LOLØ7.CBL;9

   Here the amount of extra task virtual address space n, say, specified at link time by EXTTSK=n, is replaced by an allocation of 2Ø48 words for this installed version.

$JOB

The $JOB command initiates a batch job.

FORMAT

```
$JOB [/PASSWORD:password] username job-name time-limit
```

where

password	is an alphanumeric string 1 to 6 characters long which is the user's batch password.
username	is an alphanumeric string 1 to 12 characters long which is unique to the user. The username must be a valid user-name such as one used in LOGIN.
job-name	is an alphanumeric string 1 to 12 characters long which identifies the job. The system prints the job-name at the beginning and end of the job's printed output.
time-limit	is the time-limit (in integer format) in minutes for which the batch job is to run. time-limit has a maximum value of 1440, that is, 24 hours. If this field is ommitted, the job will recieve the installation default batch time limit - usually eight minutes.

DESCRIPTION

The $JOB command must be the first command in a batch job command stream.

PASSWORD must not be specified if there is no batch password accociated with the account.

EXAMPLES

1. `$JOB PIERCE JOBONE`

2. `$JOB/PASSWORD:SECRET SYSTEM ACCOUNTS 30`

LIBRARIAN

The LIBRARIAN command allows the user to create, delete and maintain object module libraries and MACRO-11 macro libraries.

FORMAT

```
    PDS> LIBRARIAN
    OPERATION?  operation[qualifiers]
    LIBRARY? libspec <alt>
    [librarian-prompt?  text]
```

or

```
    $LIBRARIAN operation[qualifiers] filespec [text]
```

where

operation	is the librarian operation to be performed. The operations are: COMPRESS CREATE DELETE EXTRACT INSERT LIST REPLACE
libspec	is a file specification of the library file on which the operation is to be performed.
qualifiers librarian-prompt text	are all dependent on the operation specified and are described accordingly below

Library Types

There are two types of library:

- those containing object modules (object module libraries) and

- those containing macros (macro libraries).

Object module libraries are created with a default filetype of .OLB. Each object module inserted into the library has its module name (taken from the .TITLE statement) added to the module name table (MNT) and its entry points (globals) added to the entry point table (EPT).

Macro libraries are created with a default filetype of .MLB. Each macro inserted into the library has its module name (taken from the .MACRO statement) added to the module name table (MNT).

Restrictions

The following restrictions apply to the handling of object modules:

1. The size of a module is limited to 65,536 words.
2. The size of the library file is limited to 65,536 words.
3. Tables and contiguous space should be allocated the maximum anticipated size. Expanding space allocations require the COMPRESS operation to copy the entire file.
4. A fatal error results if an attempt is made to insert a module into a library which contains a differently named module with the same entry point.

COMPRESS

The COMPRESS operation physically deletes logically deleted (by the DELETE operation) modules in the file specified and re-arranges the file, putting all free space at the end of the file, where it is available for new module inserts.

Format

```
PDS> LIBRARIAN
OPERATION? COMPRESS[qualifiers]
LIBRARY? libspec
NEW LIBRARY? newlibspec
```

or

```
$LIBRARIAN COMPRESS[qualifiers] libspec newlibspec
```

where

libspec	is a specification of the library file to be compressed (no wild-cards allowed).
newlibspec	is a specification of the compressed library file (no wild-cards allowed).

The operation qualifiers are as follows:

Qualifier	Description	Default
/SIZE:n	The size in 256-word blocks of the compressed file.	100
/EPT:n	The number of entries to allocate in the entry point table (not greater than 1024). A macro library has no entry point table and then n is set to 0 even if specifically defined. n is rounded up to the nearest multiple of 64.	512 (object) 0 (macro)
/MNT:n	The number of entries to allocate in the module name table (not greater than 1024). n is rounded up to the nearest multiple of 64.	256

Examples:

1. PDS> LIBRARIAN COMPRESS/SIZE:150

   LIBRARY? PEEK.OLB <alt>

   NEW LIBRARY? PEEK2.OLB

   The object library file PEEK.OLB is compressed to 150 blocks with 512 EPT entries and 256 MNT entries by default. The compressed file is called PEEK2.OLB.

2. $LIBRARIAN COMPRESS FREAN.MLB FREAN2.MLB

   The macro library file FREAN.MLB is compressed to 100 blocks with no EPT entries and 256 MNT entries by default. The compressed file is called FREAN2.MLB

CREATE

The CREATE operation allocates a contiguous library file on a direct access device (e.g. disk), and initializes the library header and tables.

Format

```
PDS? LIBRARIAN
OPERATION? CREATE/[qualifiers]
LIBRARY? libspec
FILE? [infile-1,...infile-n]
```

or

```
$LIBRARIAN CREATE[qualifiers] libspec infile-1[,...infile-n]
```

where

libspec	is a specification of the library file to be created (no wild-cards allowed).
infile	is a specification of a file to be input to the new library file. If no infiles are supplied, an empty library file is created as the qualifiers dictate.

The operation qualifiers are as follows:

Qualifier	Description	Default
/SIZE:n	The size in 256-word blocks of the library file to be created.	100
/EPT:n	The number of entries to allocate in the entry point table (not greater than 1024). A macro library has no entry point table. n is rounded up to the nearest multiple of 64.	512(object) 0 (macro)
/MNT:n	The number of entries to allocate in the module name table (not	256

Qualifier	Description	Default
	greater than 1024). n is rounded up to the nearest multiple of 64.	
/TYPE:type	The type of library being created. type is either OBJECT or MACRO.	
/SELECT	The LINK command will use the file to define required global symbols at task build. (Object files only.)	
/SQUEEZE	Reduce the macro file by erasing all trailing blanks and tabs, blank lines and comments from the source text. (Macro files only).	
/NOENTRY_POINTS	Library modules will be stored in the library omitting definitions of the symbols that are entry points.	

Examples:

1.

```
PDS> LIBRARIAN
OPERATION?  CREATE/SI:200/EP:1024/MN:512/TYPE:OBJ
LIBRARY?  MYLIB.OLB
FILE?  ONE.OBJ, TWO.OBJ, THREE.OBJ
```

Create an object library file named MYLIB.OLB with a size of 200 blocks with 1024 EPT entries and with 512 MNT entries, from three input files.

2.

```
$LIBRARIAN CREATE/TYPE:MAC BATLIB.MAC INPUT.MAC
```

Create a macro library file named BATLIB.MAC from one input file (INPUT.MAC).

DELETE

The DELETE operation performs two kinds of deletion:

1. It deletes modules, and all their associated entry points, from the library file specified.
2. It deletes specified entries in the entry point table (EPT).

There is no restriction on the number of modules that can be deleted in one DELETE operation. If no module of the specified name exists in the library, DELETE has no effect on the library. A deleted module is marked as deleted, but remains physically in the file until a COMPRESS operation is performed.

Format

```
PDS>LIBRARIAN
OPERATION? DELETEqualifier
LIBRARY? libspec
ENTRIES? name-1[,...name-n]
```

or

```
$LIBRARIAN DELETEqualifier libspec name-1[,...name-n]
```

where

libspec	is a specification of the library file that contains the modules or entry points to be deleted.
name	is a module name or the name of an entry in the entry point table.
qualifier	is one of the following:

Qualifier	Description
/MODULES	Delete the specified module (the default qualifier).
/GLOBAL	Delete the EPT entries specified.

Examples:

1.
```
PDS> LIB DELETE/MODULES
LIBRARY? MYLIB.MLB
ENTRIES? NAMEA, NAMEB, NAMEC
```

   Delete the macros NAMEA, NAMEB and NAMEC from the macro library file MYLIB.MLB.

2.
```
$LIBRARIAN DELETE/GLOBAL  MACLIB.OLB  NAMEX
```

   Delete the EPT entry named NAMEX contained in the library file MACLIB.OLB.

EXTRACT

The EXTRACT operation extracts modules from a library and generates a new file which is the concatenation of the named modules. The original library remains unaltered.

FORMAT

```
PDS> LIBRARIAN EXTRACT/OUTPUT:filespec library module-list
```

where

filespec is the filespecification of the file to be created.

Defaults: If the output file does not have an explicit filetype, the filetype is assigned and is .MAC if the modules are extracted from a MACRO library and .OBJ if from an object library.

module-list
lists up to 8 modules to be extracted.

Example:

```
PDS> LIBR EXTR/OUT:AB MYLIB.MLB A B
```

This command causes the two modules A and B to be extracted from the MACRO library MYLIB.MLB and placed in a single file called AB.MAC.

INSERT

The INSERT operation inserts modules into the specified library file. Any number of input files are allowed any of which may contain concatenated object modules.

Format

```
PDS> LIBRARIAN
OPERATION? INSERT[qualifier]
LIBRARY? libspec
FILE? infile-1[,...infile-n]
```

or

```
$LIBRARIAN INSERT[qualifier] libspec infile-1[,...infile-n]
```

where

libspec	is a specification of the library file into which modules are to be inserted (no wild-cards allowed).
infile	is the specification of a file to be inserted into libspec.

qualifier is one of the following:

/SELECT	The LINK command will use the file to define required global symbols at task buld. (Object files only.)
/SQUEEZE	Reduce the macro-file by eliminating all trailing blanks and tabs, blank lines and comments from the source text. (Macro files only).
/NOENTRY_POINTS	Modules are inserted without the definitions of the symbols that are entry points.

Examples:

1.
```
PDS> LIBRA
OPERATION?  INSERT/SQUEEZE
LIBRARY?  MACLIB.MLB
FILE?  ONE.MAC, TWO.MAC
```

Insert the modules contained in the files ONE.MAC and TWO.MAC into the library file name MACLIB.MLB, eliminating blanks and comments.

2.
```
$LIBRARIAN INSERT MYLIB.OLB MODULE.OBJ
```

Insert the modules contained in the file MODULE.OBJ into the library file named MYLIB.OLB.

LIST

The LIST operation causes a library file directory to be printed on the user's terminal by default or to be sent to an output file. The operation qualifier also determines the amount of detail contained in the directory. By default, the directory lists all the modules in the library.

FORMAT

```
PDS> LIBRARIAN
OPERATION? LIST[qualifier]
LIBRARY? libspec
```

or

```
$LIBRARIAN LIST[qualifier] libspec
```

where

libspec	is the specification of the library file to be listed (no wild-cards allowed).
qualifier	is one of the following:

Qualifier	Description
/OUTPUT:outfile	Send the output to the specified file.
/ENTRIES	Produce a directory of all modules and list entry points for each.
/FULL	Produce a directory of all modules, giving full module descriptions: size, date of insertion and version.
/PRINT	Send the output to the lineprinter.

Examples:

1. `PDS> LIBRARIAN LIST MYLIB.MLB`

   List at the user's terminal a directory of all the modules contained in MYLIB.MLB.

2. `$LIBRARIAN LIST/FULL/OUTPUT:LP0: MODLIB.OLB`

   List at the line printer a directory of all the modules and their descriptions contained in the library file MODLIB.OLB.

REPLACE

The REPLACE operation replaces old modules in the library with new modules of the same name. That is, a new module that has the same name as a module already contained in the library replaces the existing module. The old module remains physically in the file until compressed.

Format

```
PDS> LIBRARIAN
OPERATION? REPLACE[qualifier]
LIBRARY? libspec
FILE? infile-1[,...infile-n]
```

or

```
$LIBRARIAN REPLACE[qualifier] libspec infile-1[,...infile-n]
```

where

libspec	is the specification of the library file containing the modules to be replaced (no wild-cards allowed).
infile	is the specification of a file containing the new modules (no wild-cards allowed)

qualifier is one of the following:

/SELECT	The LINK command will use the file to define required global symbols at task build. (Object files only).
/SQUEEZE	Reduce the macro file by eliminating all trailing blanks and tabs, blank lines and comments from the source text. (Macro files only.)
/NOENTRY_POINTS	Replace modules, omitting definitions of symbols that are entry points.

Examples:

1.
```
PDS> LIBRARIAN
OPERATION? REPLACE
LIBRARY? MODLIB.OLB
FILE? NEWMOD.OBJ
```

   Replace modules in the file MODLIB.OLB with modules of the same name from the file NEWMOD.OBJ.

2.
```
$LIBRARIAN REPLACE OLDLIB.OLB     ONELIB.OBJ,TWOLIB.OBJ
```

   Replace modules in the file OLDLIB.OLB with modules of the same name in the files ONELIB.OBJ and TWOLIB.OBJ.

LINK

The LINK command links object files (that is, compiled or assembled modules) to form an executable task and produces output as directed by command qualifiers.

The IAS Task Builder Reference Manual describes the Task Builder procedures and options in full; anyone using Task Builder options should first read the Task Builder manual.

FORMAT

```
     PDS>LINK[qualifiers]

     FILE? infile-1[file-qualifier][,...,infile-n]
```

or

```
     $LINK[qualifiers]   infile-1[file-qualifier][,..,infile-n]
```

where

infile	is the specification of an input file. See the section called Input Files below for further information. Wild-cards are not allowed. The user must not include this parameter if the command qualifier /OVERLAY has been specified (see the section called Command Qualifiers below)
file-qualifier	is one of the following file qualifiers. See the section called File Qualifiers for a definition of each qualifier. /CONCATENATED /LIBRARY /LIBRARY:[(]mod-1[,...,mod-n)] /NOCONCATENATED /SELECT
qualifier(s)	are one or more of the command qualifiers listed below. The section called Command Qualifiers describes each one in detail.

Qualifier	Default
/ABORT	/ABORT
/CHECKPOINT	/CHECKPOINT
/CROSS_REFERENCE	/NOCROSS_REFERENCE
/DEBUG[:filespec]	/NODEBUG
/DEFAULT_LIBRARY	/NODEFAULT_LIBRARY
/DISABLE	/DISABLE
/EXIT:n	/EXIT:1
/FIX	/NOFIX
/FLOATING_POINT	/FLOATING_POINT
/FULL_SEARCH	/NOFULL_SEARCH
/HEADER	/HEADER
/LARGE_SYMBOL_TABLE	/NOLARGE_SYMBOL_TABLE
/MAP[:filespec]	/NOMAP
/MAP:(filespec/qualifier)	/MAP:(filespec/WIDE)
/MULTIUSER	/NOMULTIUSER
/OPTIONS	/NOOPTIONS
/OVERLAY_DESCRIPTION:filespec	/NOOVERLAY_DESCRIPTION
/POSITION_INDEPENDENT	/NOPOSITION_INDEPENDENT
/PRIVILEGED	/NOPRIVILEGED
/READ_WRITE	/NOREAD_WRITE
/SEQUENTIAL	/NOSEQUENTIAL
/SYMBOLS[:filespec]	/NOSYMBOLS
/TASK[:filespec]	/TASK
/TRACE	/NOTRACE

Command Qualifiers

All the command qualifiers described in this section may be negated by the prefix NO. For example, the qualifier /TASK instructs the Task Builder to keep a task file; whereas the qualifier /NOTASK requests that a task image file should not be produced by the Task Builder.

/TASK[:filespec]

Default: /TASK

Keep a task image file.

Unless filespec is given, the task file takes the name of the first input file (or the name of the overlay descriptor file) except that the filetype is TSK.

If filespec is given, the filetype field may be omitted; in which case, the Task Builder assumes it to be TSK.

/MAP[:filespec] or /MAP[:(filespec/qualifier)]

Default: /NOMAP

Produce a memory allocation map.

If filespec is not specified after /MAP, the map file is sent to the line printer.

If filespec is given, the filetype field may be omitted; in which case, the Task Builder assumes it to be MAP.

The following qualifiers can be attached to the map filespec:

/FULL	Include all modules in map
/FILES	Include file-by-file breakdown
/NARROW	Make map in 72-column format
/SHORT	Make only summary of map
/WIDE	Make map in 132-column format

Defaults: /NOFULL /NOFILES /WIDE

/SYMBOLS[:filespec]

Default: /NOSYMBOLS

Produce a symbol table file.

Unless filespec is given, the symbol table file takes the name of the first input file, except that the filetype is STB.

If filespec is given, the filetype field may be omitted; in which case, the Task Builder assumes it to be STB.

/OPTIONS

Default: /NOOPTIONS

Apply Task Builder options specified after the command string.

In interactive mode, the /OPTIONS qualifier causes the Task Builder to prompt "OPTIONS?" after the input files have been specified.

For example:

```
     PDS> LINK/OPTIONS

     FILE? PROG REPORT

     OPTIONS?
```

The user then enters the options which are described in the list below. A slash (/) as the first character in a line then terminates the list of options and the Task Builder begins executing. Details of individual option syntax are contained in the IAS Task Builder Reference Manual.

For example:

```
     PDS> LINK/OPTIONS

     FILES? MAIN.OBJ, PROG.OBJ

     OPTIONS? ACTFIL=8

     OPTIONS? MAXBUF=160

     OPTIONS? UNITS=9

     OPTIONS? ASG=DT1:7:8:9

     OPTIONS? /
```

In batch mode, the presence of the /OPTIONS qualifier in the command qualifier list causes the Task Builder to expect one or more options to be specified on lines immediately following the command string.

A line containing a slash (/) in the first character position terminates the list of options.

The letters F and M in the list of Task Builder options below indicate for which language, FORTRAN or MACRO, the option is relevant. Those marked F apply also to CORAL, except where a C (for CORAL) is shown explicitly.

Option	Meaning	
ABSPAT	Declare absolute patch values.	M
ACTFIL	Declare number of files open simultaneously.	FM
ASG	Declare device assignment to logical units.	FM
BASE	Define lowest virtual address.	FM
COMMON	Declare task's intention to access a (read/write) shareable global area.	FM
EXTSCT	Declare extension of a program section.	FM

Option	Meaning	
EXTTSK	Extend task memory allocation at install time.	FM
FMTBUF	In FORTRAN, declare extension to buffer used for processing format strings at run-time.	F
	In CORAL, set to blkmax*8, where blkmax is the maximum number of LUNs used for concurrent asychronous block I/O at any one time.	C
GBLDEF	Declare a global symbol definition.	M
GBLPAT	Declare a series of patch values relative to a global symbol.	M
LIBR	Declare task's intention to access a (read-only) shareable global area.	FM
MAXBUF	In FORTRAN, declare an extension to the FORTRAN record buffer.	F
	In CORAL, set to strmax*140 (decimal), where strmax is the maximum number of LUNs associated with stream I/O at any one time.	C
ODTV	Declare the address and size of the debugging aid SST vector.	M
PAR	Declare partition name and dimensions.	FM
POOL	Declare pool usage limit.	FM
PRI	Declare priority.	FM
RESCOM	Declare task's intention to access a shareable global area held in the specified user file directory.	FM
RESLIB	Declare task's intention to access a shareable global area held in the specified user file directory.	FM
STACK	Declare the size of the task's stack.	FM
SYMPAT	Declare a patch using task symbols.	M
TASK	Declare the default installed name of the task.	FM
TOP	Define highest virtual address.	FM
TSKV	Declare the address of the task SST vector.	M
UIC	Declare the user identification code under which the task runs.	FM
UNITS	Declare the maximum number of logical units.	FM

/OVERLAY_DESCRIPTION:filespec

Default: /NOOVERLAY_DESCRIPTION

Link the task according to the overlay structure defined in the given file, the name of which must be included with the /OVERLAY_DESCRIPTOR qualifier. If the filetype field of filespec is omitted, the Task Builder assumes it to be ODL.

The input files to LINK are specified within the overlay description file; therefore they must not be specified in the input file parameter list.

See the IAS Task Builder Reference Manual for details of ODL files.

/DEBUG[:filespec]

Default: /NODEBUG

If filespec is not given, link the task with the system's debugging aid. If filespec is given, link the task with the debugging aid contained in the specified file. The debugging aid must be in object format.

/ABORT

Default: /ABORT

The task can be aborted.

/CHECKPOINT

Default: /CHECKPOINT

The task can be checkpointed.

/CROSS_REFERENCE

Default: /NOCROSS_REFERENCE

Append a global symbol cross-reference to the end of the memory allocation map.

/DEFAULT_LIBRARY:file-spec

Default: /DEFAULT_LIBRARY:LB:[1,1]SYSLIB.OLB

Use the named object module library instead of current system library file LB:[1,1]SYSLIB.OLB

/DISABLE

Default: /DISABLE

The task can be disabled.

/EXIT:n

Default: /EXIT:1

Task Builder stops executing after n(decimal) errors.

/FIX

Default: /FIX

The task can be fixed in memory.

/FLOATING_POINT

Default: /FLOATING_POINT

The task uses the floating point processor.

/FULL_SEARCH

Default: /NOFULL_SEARCH

This controls symbol table searching in overlaid tasks having co-trees.

/HEADER

Default: /HEADER

The task includes a header. /NOHEADER should be used when producing a non-executable task image, for example a library or common shareable global area.

/LARGE_SYMBOL_TABLE

Default: /NOLARGE_SYMBOL_TABLE

Select a version of the Task Builder that has a large internal symbol table. (Considerably slower than the default Task Builder.)

/MULTIUSER

Default: /NOMULTIUSER

The task is multiuser.

/POSITION_INDEPENDENT

Default: /NOPOSITION_INDEPENDENT

The task code is position independent.

/PRIVILEGED

Default: /NOPRIVILEGED

The task is 'executive privileged'.

/READ_WRITE

Default: /NOREAD_WRITE

Give Read/Write access to the Read-Only code.

/SEQUENTIAL

Default: /NOSEQUENTIAL

Program sections within the task are to be linked in the order in which they first appear. Otherwise they are linked in alphabetical order.

/TRACE

Default: /NOTRACE

The task is traceable.

Input Files

Input files to the LINK command may be specified in one of two ways:

1. In a list of file specifications as a parameter to the command.

2. From within an overlay description file by means of the /OVERLAY command qualifier.

If the /OVERLAY qualifier has been used to specify the input files, they must not also be specified as a command parameter (see item 1 above). The input files may consist of:

1. Single object modules

2. Concatenated object modules

3. Object module libraries

4. Symbol table files

File qualifiers must be used to identify concatenated module files and library files (see the section called File Qualifiers below). In addition, the /SELECT qualifier may modify symbol table files; the Task Builder then uses the modified file only to resolve required symbol definitions.

The Task Builder provides default filetypes in the following cases. When specifying single or concatenated object modules, the user may omit the filetype field. The Task Builder then assumes the filetype to be .OBJ. The filetype field of a library file (a file modified by the /LIBRARY qualifier) may also be omitted, in which case the Task Builder assumes the filetype to be OLB.

Symbol table files, however, have no default filetype, so the filetype field must be supplied.

Wild-cards are not allowed for any type of file specification supplied with LINK.

File Qualifiers - The following list defines all the available file qualifiers.

File Qualifier	Description
/CONCATENATED	Identifies the file as a concatenated object file.
/LIBRARY	Identifies the file as an object module library file.
/LIBRARY:[(]mod-1[,...,mod-n)]	Identifies the file as an object module library file where mod is the name of an object module and instructs the Task Builder to take only the modules named.
/NOCONCATENATED	Instructs the Task Builder to take only the first module in the file. If it is a concatenated object module file, subsequent modules are ignored.
/SELECT	Instructs the Task Builder to take only required global symbol definitions from the file. The modified file may be any object file, but it is normally a symbol table file.

EXAMPLES

1.
```
$LINK/OPTIONS/PRIVILEGE A.OBJ/CONCATENATED
UNITS=9
/
```

2.
```
PDS> LINK/OVERLAY:STRUCTURE/MAP:ROUTE
```

   The system does not prompt FILE? if /OVERLAY has been specified.

3.
```
PDS> LINK/DEFAULT_LIBR:DK1:[1,1]SYSLIB

FILE? A.OBJ, B.OBJ
```

LOGIN

The LOGIN command initiates an interactive session at a terminal.

FORMAT

```
     PDS> LOGIN [/qualifiers]

     USERID? username

     PASSWORD?  password
```

where

qualifiers are either of the following:

/NONOTICE	to suppress the notice message that, if previously set up, is automatically printed at login.
/QUIET	to suppress certain non-critical system information (for example, accounting information).

username	is an alphanumeric character string 1 to 12 characters long which is unique to the user.
password	is an alphanumeric character string 1 to 6 characters long associated with the user's username. As a security measure, the system does not print the password when it is entered in response to the PASSWORD? prompt.

The username and password are supplied to the user by the system manager.

DESCRIPTION

The LOGIN command is usually the first command issued by the interactive user (after the initial CTRL/C).

EXAMPLES

1.
```
     PDS> LOGI  JOHNDOE

     PASSWORD?  secret

     PDS>
```

2.
```
PDS> LOGIN

USERID? MONTY

PASSWORD?  python

PDS>
```

3.
```
PDS> LOGI/NONOT MKEE

PASSWORD?  carlsb

PDS>
```

LOGOUT

The LOGOUT command terminates the user's interactive session and releases any allocated devices and mounted volumes.

FORMAT

```
PDS> LOGOUT
```

The LOGOUT command has no parameters.

DESCRIPTION

The LOGOUT command causes the system to display the following information if "QUIET" mode has not been set (see PDS> SET QUIET):

1. The volumes and devices deallocated and dismounted
2. The user's username, UIC, terminal number and Job-id.
3. The logout time
4. The connect time
5. CPU utilization

If PDS> SET PRINTING DEFERRED is in force, any spooled files generated by tasks run from the user's terminal are printed when the user logs out.

The message BYE then appears and indicates that the terminal is inactive.

EXAMPLE

```
PDS> LOGOUT


BYE
```

MACRO

The MACRO command assembles one or more ASCII source files containing MACRO-11 statements into a single relocatable binary object file. The output optionally consists of a binary object file, an assembly listing, a cross-reference listing and the symbol table listing.

FORMAT

```
     PDS> MACRO[qualifiers]

     FILE? filespec[/LIBRARY][+...+filespec]
```

or

```
     $MACRO[qualifiers]  filespec[/LIBRARY][+...+filespec]
```

where

filespec	is the specification of a file that contains MACRO source code. Multiple input file specifications must be concatenated with a plus sign (+). No wild-cards are allowed. Specifications must include a filename. If the filetype is omitted, the system assumes it to be MAC.
/LIBRARY	if present, indicates that the file is a macro library file. A user macro library file must be specified in the command line prior to the source files that reference the library.
qualifiers	to the MACRO command are one or more of the following:

Qualifier	Meaning
/OBJECT[:filespec]	Produce an object file (the default condition), named accordingly if filespec (no wild-cards) is supplied. Otherwise the file is named by default (see Defaults below).
/NOOBJECT	Do not produce an object file.
/LIST[:filespec]	Produce a listing file (the default is /NOLIST), named accordingly if filespec is supplied. Otherwise the file is named by default (see Defaults below).
/NOLIST	Do not produce a listing file.
/CROSS_REFERENCE	Append to the assembly listing a cross-reference of user symbols and macro symbols referenced in the source files. For further control, see MACRO SWITCHES below. (Default: /NOCROSS)
/SWITCHES:(swlist)	Use the list of switches 'swlist' to control the contents or format of the output files. See MACRO SWITCHES, below. (Default: /NOSWITCHES)

Defaults

Object File - By default the assembler produces an object file with the name of the last source file specified and .OBJ as the filetype.

Listing File - A listing file is sent to the line printer if /LIST is specified with no filename. If filespec is defined without a filetype then .LST is assumed.

MACRO SWITCHES

Some MACRO switches are available via the /SWITCHES:(swlist) qualifier. swlist can include one or more of /LI (list), /NL (do not list), /CR (cross reference).

/LI and /NL can be followed by the following switch values, separated from /LI or /NL and from each other by colons.

Value	Default	Items Listed (/LI) or Not Listed (/NL)
BEX	list	binary extensions
BIN	list	generated binary code
CND	list	unsatisfied conditional coding
COM	list	comments
LD	no list	listing directives that alter the listing level count
LOC	list	location counter
MC	list	macro calls and repeat expansions
MD	list	macro definitions and repeat expansions
ME	no list	all macro expansions
MEB	list	only macro expansions that generate binary code
SEQ	list	sequence numbers of source lines
SRC	list	source lines
SYM	list	symbol table of assembled source program
TOC	list	table of contents during assembly pass 1
TTM		/LI:TTM 80-column output /NL:TTM 132-column output default: installation-dependent

/CR can be followed by the following switch values, separated from /CR and from each other by colons.

Value	Default	Symbols Cross Referenced
SYM	list	user defined symbols
MAC	list	macro symbols
PST	no list	permanent symbols
REG	no list	register symbols

If one or more values are specified, only the corresponding types of symbol are cross-referenced. The switch /CR cannot be used in conjunction with the command qualifier /CROSS_REFERENCE.

COMMENTS

For further information on the use of MACRO-11, refer to the IAS/RSX-11 MACRO-11 Reference Manual.

EXAMPLES

1.
```
PDS> MACRO
FILE? A.AMC+B.MAC;3
```

2.
```
$MACRO/NOLIST FILEA.MAC
```

3.
```
PDS> MAC/OBJ:C.OBJ    D.MAC+E.MAC
```

4.
```
PDS> MAC MYFILE.MAC
```

5.
```
PDS> MAC/LIST MACLIB.MLB/LIB+MYFILE
```

6.
```
PDS> MAC/NOOBJ/LI/SW:(/LI:ME/CR:SYM:MAC:REG) TEST.MAC
```

7.
```
PDS> MAC/LI:FILE/SW:(/LI:TTM) TEST
```

MERGE

The MERGE command takes records from a SEQUENTIAL, INDEXED or RELATIVE file (the transaction file) and merges them with an INDEXED or RELATIVE file (the target file).

FORMAT:

```
PDS> MERGE[/LOG[:filespec]] transactionfile[/qual1] targetfile/qual2
```

where

/LOG — if specified sends an error log to filespec or by default to the user's terminal. The log gives details of records that could not be merged.

/qual1 — is one of:

- /SEQUENTIAL
  transaction file is sequential
- /INDEXED [/KEY:NUMBER:n]
  transaction file is an Indexed Sequential (ISAM) file. The order of record extraction can be specified by the /KEY qualifier and key number.

  Default: /KEY:NUMBER:1 (the primary key).

  /INDEXED may be omitted if /KEY:NUMBER:n is specified.
- /RELATIVE specifies a relative structured file.

qual2 — must be specified and is either

/INDEXED

or

/RELATIVE

MESSAGE

The MESSAGE command sends a specified message to the operator's reporting terminal.

FORMAT

```
     PDS> MESSAGE

     MESSAGE?  message
```

or

```
     $MESSAGE message
```

where

message	is a string of 1- to 65-characters terminated by carriage return in interactive mode, or a string written on the same line as the $MESSAGE command in batch.

EXAMPLE

```
     $MESSAGE THIS JOB WILL REQUIRE 2 TAPE DRIVES
```

MOUNT

The MOUNT command makes a volume available to the user and optionally associates a logical name with the volume.

FORMAT

```
     PDS> MOUNT[qualifier]

     DEVICE?  device-name

     VOLUME-ID?  volume-identification

     [LOGICAL NAME?  logical-name]
```

or

```
     $MOUNT[qualifier(s)] device-name volume-id logical-name
```

where

qualifier(s) — is one or more qualifiers, most of which may only be specified when a volume is initially mounted. See the section Command Qualifiers below.

device-name — is the device or the logical name of the device on which the volume is to be mounted. The device unit number may be omitted, except when the /NOOPERATOR qualifier is used or the device name is a logical name.

NOTE

The system will not prompt for 'logical-name' if the device was mounted by the logical name that was assigned to it by an ALLOCATE command.

volume-identification — is the volume identification written in the volume's header. If the volume is being mounted as 'FOREIGN', or if the qualifier /OVERRIDE:volume is used, then the name supplied here is that which identified the volume for handling by the operator, such as a label written on the volume container. For disk and DECtape the volume identification is 1 to 12 characters long. For ANSI labelled magnetic tape the identification (ANSI label) is 1 to 6 characters long.

logical-name — is the logical name to be associated with the physical device.

DESCRIPTION

The MOUNT command is normally used to make a specified volume available to a timesharing user. It may also be used to mount a volume globally or for realtime purposes only. A globally mounted volume is potentially available to all timesharing users and is only fully dismounted when an explicit DISMOUNT/GLOBAL command is issued (from any PDS terminal). A volume mounted for real-time allocates the device for real-time purposes only and so cannot be accessed by timesharing tasks until the owner issues an explicit DISMOUNT/REALTIME command.

The user obtains exclusive access to magnetic tape volumes and to any volumes mounted as foreign. Files-11 disk and DECtape volumes may be shared; that is, once the volume has been mounted, other users may also use it.

The unit number will normally be omitted from the device specification. The system then selects the appropriate unit. The MOUNT command may be qualified in the following circumstances:

1. When a specified Files-11 disk or DECtape volume is not already mounted in the system.

2. When the user mounts a magnetic tape or foreign volume.

Command Qualifiers

The system ignores command qualifiers if the command is mounting a previously mounted Files-11 disk or DECtape.

* Qualifiers marked with an asterisk allow the first user to override parameters set when the volume was initialized.

Qualifier	Description
*/ACCESSED:n	Number of preaccessed directories to be kept (Files-11 disk and DECtape only).
*/DENSITY:n	Set magnetic tape density where n = 8ØØ or 16ØØ
*/EXTENSION:n	Set default file extension to n blocks.
*/FILE_PROTECTION:(code)	Override default protection code to be given to new files. (See Chapter 6, Section 6.1.3)
/FOREIGN	Allocate the volume as foreign (that is, single user). The default is Files-11 format. This qualifier cannot be specified with /GLOBAL (see below).
/GLOBAL	The volume is to be mounted globally.
/NOOPERATOR	Mount without operator intervention. The device unit number is mandatory.
/NOWRITE	Write protected; that is, the volume may not be written to. Default is write permitted.
/OVERRIDE:(items)	where items are one or more of the following.

Parentheses may be omitted if only one item is specified.

EXPIRATION	allows the user to over-write an unexpired magnetic tape volume.
SET_IDENTIFICATION	allows the user to process tapes with inconsistent file set identifiers.
VOLUME_IDENTIFICATION	allows the user to override the volume identification, thus allowing the user to mount specifying any label that identifies the volume (for example, a label written on the volume container).

Qualifier	Description
/PROTECTION:(code)	Replace volume protection with code specified. (See Chapter 6, Section 6.1.3)
/REALTIME	Mount volume for access by realtime tasks only.
/UNLOCKED	Leave index file unlocked (Files-11 disk and DECtape only). Default is to leave index file locked.
/NOSHARE	Mount a Files-11 volume for exclusive use.
/DEVICES:n	Allocate the stated number of device units for a multi-volume magnetic tape unit set.

/PROCESSOR:ACPtask

Specifies the Ancillary Control Processor (ACP) to be used for processing file accesses to the volume. The ACP specified by this qualifier will override the default ACP.

EXAMPLES

```
     1.  PDS> MOUNT
         DEVICE? DT2:
         VOLUME-ID? RISE <CR>

     2.  $MOUNT/FOREIGN MT: TESTER CFØ:

     3.  PDS> MOU DK:
         VOLUME-ID? SAM AL1:

     4.  $MOUNT/DEN:8ØØ/NOOPER MTØ: VOL163 TAØ:

     5.  PDS> ALLOC DEVICE
         DEVICE? DT <ALT>
         LOGICALNAME? XX
         PDS> MOU/FOR XX DOSVOL2
```

ON

The ON command is used only in an Indirect Command File or Batch Command File. ON controls the processing of such a file after the completion of any command-line that returns an error status to PDS.

FORMAT

```
     [$]ON error-severity action
```

Where

```
     error-severity  is one of

                          WARNING
                          ERROR
                          SEVERE_ERROR

     action          is one of

                          CONTINUE

                          GOTO label

                               label is an alphanumeric string and must
                               appear together with a colon in front of
                               a later command in the file.

                          STOP

                          any fully specified PDS command
```

An ON command must be entirely specified on one line.

DEFAULT

[$]ON ERROR STOP is assumed by default at the beginning of a terminal session (LOGIN) or the beginning of a batch job ($JOB). If an ON statement is found, on attempted execution, to be itself faulty, PDS reverts to the default setting.

DESCRIPTION

ON takes effect only after completion of one or more subsequent lines in the command file. An ON command remains in force until the next ON command and is then superseded entirely. See this manual, Sections 8.5 and 8.5.1 for a description of ON and associated commands.

EXAMPLES

1.
```
$ON ERROR STOP
$MACRO MYPROG
$LINK MYPROG
$RUN MYPROG
```

   Here $ON has no effect on the MACRO assembly itself. If the assembly is completed with nothing worse than a warning, the job proceeds to $LINK. If the linking is completed with nothing worse than a warning, the job proceeds to $RUN.

2.
```
$JOB ENGINE3
$ON WARNING GOTO ELSE
$LINK MYPROG
$RUN MYPROG
$STOP
$ELSE:  LINK OLDPROG
$RUN OLDPROG
$EOJ
```

PRINT

The PRINT command causes one or more specified files to be queued for output on the line printer. The user may optionally delete the file or files after they have been printed.

FORMAT

```
     PDS> PRINT[/DELETE][/FORMS:n][/COPIES:n][/PRIORITY:n]-
     [/NOBANNERS][/NOTRANSFER]

     FILE? filespec-1[,...filespec-n]
```

or

```
     $PRINT[/DELETE][/FORMS:n][/COPIES:n][/PRIORITY:n][/NOBANNERS]-
     [/NOTRANSFER] filespec-1[,...filespec-n]
```

where

/DELETE instructs the system to delete the file or files after they have been printed.

/FORMS:n (where n is a digit from 0 to 6) indicates the type of form on which the specified files are to be printed. The association of a value of n with a particular form is installation dependent.

Default: /FORMS:0

/COPIES:n (where n is an integer from 1 to 32) determines the number of file copies to be printed.

Default: /COPIES:1

/PRIORITY:n
allows a user to request that a file be printed at a low priority (for example, priority 1). n must be between 1 and 100.

/NOBANNERS
suppresses the printing of the file identification banner pages.

/NOTRANSFER
inhibits the copying of the queued file(s) to the spooling device. The file(s) will be printed direct from the volume on which it resides.

filespec is the specification of a file to be printed. Wild-cards are allowed. The filetype is optional and is defaulted to LST.

DESCRIPTION

The specified file or files are submitted to the line printer and subsequently deleted if the user has included the /DELETE qualifier. If files are queued with more than one value of /FORMs, a message is sent to the operator when a change of forms type becomes necessary so that the remainder of the queues may be output.

EXAMPLES

1.
```
PDS> PRINT

FILE?  MACLIST
```

2.
```
$PRINT FREAN.MAC;3, PEEK.CAF;*
```

3.
```
PDS> PRI/DE B4.FAL
```

QUEUE

The QUEUE command allows the user to access the queue in the following ways:

1. To interrogate the queue (/LIST)
2. To remove an entry belonging to the user from the queue (/REMOVE)
3. To add to the queue (/ADD)

Note that the simpler commands PRINT and SUBMIT should be used to add files to the line printer and batch queues.

FORMAT

The format of the command depends on the queue operation to be performed.

The default operation is /ADD.

LIST

```
     PDS> QUEUE/LIST
```

Description

Display the status of the user's queue entries.

REMOVE

```
     PDS> QUEUE/REMOVE

     SEQUENCE? seqno
```

where

seqno	is the sequence number of a queue entry to be removed, determined by issuing a QUEUE/LIST command.

Description

Remove the queue entry specified by a sequence number, which is displayed via the QUE/LIST command described above.

ADD

```
PDS> QUEUE/ADD[/FORMS:n][/COPIES:n][/DELETE][/PRIORITY:n]-
[/NOBANNERS][/NOTRANSFER]

QUEUE? device-name

FILE? filespec
```

where

/FORMS:n	(where n is a digit from Ø to 6) indicates the type of form on which the specified files are to be printed. The association of a value of n with a particular form type is installation dependant. Default: /FORMS:Ø
/COPIES:n	(where n is an integer from 1 to 32) determines the number of copies to be printed. Default: /COPIES:1
/DELETE	requests the system to delete the specified files after they have been processed.
/PRIORITY:n	allows a user to request that a file be queued at a low priority. n must be between 1 and 1ØØ.
/NOBANNERS	suppresses the printing of the file identification banner pages.
/NOTRANSFER	inhibits the copying of the queued file to the spooling device. The file will be printed direct from the volume on which it resides.
device-name	specifies the relevant queued device.
filespec	is the specification of a file to be added to the queue specified. Only one filespec is allowed. It must contain a filename and filetype. Wild-cards are allowed. The filetype is optional and is defaulted to LST.

Description

Add the specified file to the named queue and, optionally, modify the resultant operation according to any specified qualifiers.

EXAMPLES

```
     1.  PDS> QUEUE/LIST

     2.  PDS> QUEUE/REMOVE  2

     3.  PDS> QUEUE/ADD/COPIES:4/DELETE
         QUEUE? LPØ:
         FILE? LIST.MAP;4

     4.  PDS> QUEUE/PRIO:1Ø LP3 LISTFILE

     5.  $QUEUE/PRIORITY:4Ø LP3 ADD.MAC
```

REMOVE

The REMOVE command allows the user to remove an installed task or a Shareable Global Area from the system. Removing a task or SGA undoes the effect of the INSTALL command. A task cannot be removed if that task is active, fixed, or has nodes accounted to it. If there is any outstanding data from SEND directives to the task it is returned to the pool.

A shareable global area cannot be removed until all tasks which map on to it have been removed.

FORMAT

```
     PDS> REMOVE[/qualifier]
     TASK?  name
```

where

/qualifier	is one of /COMMON /LIBRARY
name	is the installed name of the task, common area or library being removed.

EXAMPLES

1. `PDS> REMOVE MYLOL`

   Remove the task with installed task name MYLOL.

2. `PDS> REM/COM SYST2Ø`

   Remove the SGA with installed name SYST2Ø.

RENAME

The RENAME command renames an existing file.

FORMAT

```
     PDS> RENAME

     OLD? oldspec

     NEW? newspec
```

or

```
     $RENAME oldspec, newspec
```

where

oldspec	is the specification of an existing file.
newspec	is the new name for oldspec.

DESCRIPTION

Both oldspec and newspec must contain a file name and filetype. Wild-cards are allowed. The device field in both file specifications must be the same because files cannot be renamed from one device to another. If the version field is omitted, the normal defaults apply (see Chapter 6, Section 6.2.1).

EXAMPLES

```
    1.   PDS> RENAME

         OLD? MYFILE.OBJ;1

         NEW? BACKUP.OBJ;1

    2.   $RENAME MYFILE.OBJ;1,BACKUP.OBJ;1

    3.   PDS> RENAME

         OLD? MYFILE.OBJ;1,BACKUP.OBJ;1

    4.   PDS> RENAME CAROL.*;*
         NEW? FRED.CBL;*
```

RUN

The RUN command causes an executable task to execute.

RUN can be issued for a timesharing task (FORMAT 1) or for a realtime task (FORMAT 2 through 6).

If a real time task is to be run, then:

1. It must already have been installed in the system (see the INSTALL command).
2. The user can take the opportunity to reset the task's UIC, partition and priority from those in force at installation.
3. The user can suppress the PDS prompt, for example to allow a terminal dialogue with the task. CTRL/C will reactivate the PDS prompt. Under /NOPROMPT PDS is still running and will timeout in the usual way if CTRL/C is not typed.

NOTE

In general, in the command format for a realtime task, taskname refers to the installed taskname (see INSTALL). This is not necessarily the same as the filename of the task image file.

For real time applications, the RUN command has one of the following basic formats:

Format 2 Request that a task be run as soon as memory is available and optionally reschedule the task to be run periodically (/REALTIME).

Format 3 Run a task immediately (/MEMORY).

Format 4 Synchronize the running of a task with a time unit and, optionally, reschedule the task after a specified interval (/SYNCHRONIZE).

Format 5 Schedule the task for running at a specified future time, and, optionally, reschedule the task after a specified interval (/SCHEDULE).

Format 6 Delay the task for a specified period and, optionally, reschedule the task to rerun periodically (/DELAY).

FORMAT 1 (timesharing)

```
     PDS> RUN

     FILE? filespec
```

where

filespec — is the specification of a file that contains an executable task. The specification must include a file name. If the filetype field is omitted, TSK is assumed.

DESCRIPTION

This form of RUN causes an executable timesharing task to execute.

To suspend an executing task run interactively, the user types CTRL/C. The user may either type CONTINUE to resume task execution or ABORT to abort the task.

Executing tasks that were submitted to the batch queue cannot be suspended.

EXAMPLES

```
     1.   PDS> RUN [200,40]PASCAL.TSK;4

     2.   $RUN PASCAL
```

FORMAT 2

```
     PDS> RUN/REALTIME[/INTERVAL:interval][/options]
     TASK?  taskname
```

where

options — are any of the following:

/UIC:[m,n]	[m,n] is User Identification Code
/PARTITION:par	par is partition name
/PRIORITY:pri	pri is priority number (decimal)
/NOPROMPT	suppresses PDS prompt

interval — is the time interval at which the task is to be periodically rerun, of the form:

```
xxt
```

where

xx — is the number of hours, minutes, seconds or ticks

t — is one of the following:

```
                              H for hours
                              M for minutes
                              S for seconds
                              T for clock ticks


     taskname        is the name of the task  to  be  run  as  soon  as
                     memory is available.
```

EXAMPLES

```
     PDS> RUN/REALTIME/UIC:[30,11] SCAN2

     PDS> RUN/REALT/PRI:120 MART9
```

FORMAT 3

```
     PDS> RUN/MEMORY[/options]
     TASK?  taskname
```

where

```
     options         are any one of the following:
                     /UIC:[m,n]      [m,n] is User Identification Code
                     /PARTITION:par  par is partition name
                     /PRIORITY:pri   pri is priority number (decimal)
                     /NOPROMPT       suppresses PDS prompt


     taskname        is the name of the task to be run immediately.  If
                     sufficient memory to run the task is not available
                     an error message is returned.
```

EXAMPLES

```
     PDS> RUN/MEMORY/PART:FILE JK03

     PDS> RUN/MEM/UIC:[100,10] MART6
```

FORMAT 4

```
PDS> RUN/SYNCHRONIZE:unit[/DELAY:delay][/INTERVAL:interval][ /options]

TASK?  taskname
```

where

```
     unit            is the synchronization clock unit, as follows:

                          HOURS for hours
                          MINUTES for minutes
                          SECONDS for seconds
                          TICKS for clock ticks

     delay           is the delay period after synchronization, of  the
                     form:
```

	xxt as in FORMAT 2
interval	is the time interval at which the task is to be periodically rerun, also of the form:
	xxt as in FORMAT 2
options	are any of the following:
	/UIC:[m,n] — [m,n] is User Identification Code
	/PARTITION:par — par is partition name
	/PRIORITY:pri — pri is priority number (decimal)
	/NOPROMPT — suppresses PDS prompt
taskname	is the name of the task to be synchronized

EXAMPLES

1. `PDS> RUN/SYNC:HOUR/DELAY:1ØM/INTERVAL:3ØM CAROL`

   When the time is next an exact number of hours, wait ten minutes, then run task CAROL every twenty-five minutes.

   If the time is now 1Ø.15, then task CAROL runs at 11.1Ø, 11.35, 12.ØØ, 12.25 and so on.

2. `PDS> RUN/SYNCH:HOUR/DELAY:5M/PART:SYSTEM XK3`

   Run task XK3 at 5 minutes past the next hour in the SYSTEM partition.

FORMAT 5

```
PDS> RUN/SCHEDULE:time[/INTERVAL:interval][/options]
TASK? taskname
```

where

time	is the absolute time of day the task is to begin execution. Time is expressed as hh:mm:ss
interval	is the time interval at which the task is to be periodically rerun, also of the form:
	xxt as in FORMAT 2
options	are any of the following:
	/UIC:[m,n] — [m,n] is User Identification Code
	/PARTITION:par — par is partition name
	/PRIORITY:pri — pri is priority number (decimal)
	/NOPROMPT — suppresses PDS prompt
taskname	is the name of the task to be scheduled

EXAMPLES

1.
```
PDS> RUN/SCHED:10:23:00/INTER:30S MKLOL
```
Run task MKLOL at 10:23:00 and every 30 seconds thereafter.

2.
```
PDS> RUN/SCHED:10:30:00/PRI:120 MYTSK
```
Run MYTSK at 10:30:00 at priority 120.

FORMAT 6

```
PDS> RUN/DELAY:delay[/INTERVAL:interval][/options]
TASK? taskname
```

where

delay	is the delay period before the task is to be periodically rerun, of the form xxt as in FORMAT 2
interval	is the time interval at which the task is to run after a delay, specified as xxt as in FORMAT 2
options	are any of the following:
taskname	is the name of the task to be run after delay.

Options:

/UIC:[m,n]	[m,n] is User Identification Code
/PARTITION:par	par is partition name
/PRIORITY:pri	pri is priority number (decimal)
/NOPROMPT	suppresses PDS prompt

EXAMPLES

1.
```
PDS> RUN/DELAY:30M/INTERVAL:20S/UIC:[30,2] MYTSK
```
Wait 30 minutes, then run [30,2]MYTSK every 20 seconds.

2.
```
PDS> RUN/DELAY:2H/PART:GLOBZ XKEE9
```
Wait 2 hours, then run task XKEE9 in the partition GLOBZ.

SET

The SET command is used for the following:

1. To suppress the output of certain information messages (SET QUIET). See FORMAT 1.
2. To establish a new default device or UFD or both for subsequent file specifications supplied by the user (SET DEFAULT). See FORMAT 1.
3. To defer printing of spooled files (SET PRINTING DEFERRED). See FORMAT 1.
4. To change the user's interactive password (SET PASSWORD). See FORMAT 2.
5. To change the user's batch password (SET PASSWORD/BATCH) See FORMAT 3.
6. To change the characteristics of the user's terminal. Users logged in under a username whose UIC is [1,1] can change the characteristics of any terminals (SET TERMINAL). See FORMAT 4, FORMAT 5.
7. To change the protection code of a file (SET PROTECTION). See FORMAT 6.
8. To reset the priority of an active task (SET PRIORITY). See FORMAT 7.

FORMAT 1

```
     PDS> SET

     FUNCTION?  parameter
```

or

```
     $SET parameter
```

where

parameter is one of the following:

QUIET or NOQUIET	Suppress or allow the output of informative (usually accounting) messages. The system default is SET NOQUIET.
DEFAULT [device-name:][ufd]	Change the user's default device and/or UFD to the value or values specified. If both device-name and ufd are omitted, the system reestablishes the user's initial default settings for both values.
PRINTING DEFERRED	Defer the printing of spooled files generated by the timesharing tasks run from the user's

terminal. This holds good until either the user logs out (by choice or timeout) or the user issues PDS> SET PRINTING NODEFERRED

PRINTING NODEFERRED
This is the normal system default.

DESCRIPTION

Changing Defaults

The system manager allocates a default device to each user, which is in effect when the user logs in. The initial default UFD is equivalent to the user's UIC. The user must issue the SET DEFAULT command to change either or both values for file specifications included in subsequent commands. The command does not affect file specifications written in programs. To reestablish the default settings in effect at login, the user issues SET DEFAULT without any other values.

EXAMPLES

```
1.  PDS> SET QUIET

2.  $SET DEFAULT [30,3]

3.  PDS> SET DEFAULT DK0:

4.  PDS> SET PRINTING DEFERRED
```

FORMAT 2 (SET PASSWORD)

```
PDS> SET PASSWORD

OLD PASSWORD? oldpassword

NEW PASSWORD? newpassword
```

where

oldpassword	is the 1- to 6-character alphanumeric password currently associated with the user's username.
newpassword	is the 1- to 6-character alphanumeric password that supersedes the old password.

DESCRIPTION

The system does not display either the old or the new password. This command is not permitted in batch mode.

EXAMPLE

```
    PDS> SET PASSWORD

    OLD PASSWORD?  glove

    NEW PASSWORD?  mitten
```

FORMAT 3 (SET PASSWORD/BATCH)

This command allows the user to re-define or define the batch password to be associated with his account. Until this command is issued, any user can submit a batch job that could run for and be charge to the users account. This command is not permitted in batch mode.

```
    PDS> SET PASSWORD/BATCH

    OLD PASSWORD?  oldbatchpassword

    NEW PASSWORD?  newbatchpassword
```

where

oldbatchpassword	is a 1- to 6-character alphanumeric string already associated with the user's username. This reply is ignored if a batch password did not already exist for that user.
newbatchpassword	is a 1- to 6-character alphanumeric string to replace the original batch password.

EXAMPLE

```
    PDS> SET PASSWORD/BATCH

    OLD PASSWORD?  sunday

    NEW PASSWORD?  monday
```

FORMAT 4 (SET TERMINAL)

The PDS> SET TERMINAL command allows the user to change the characteristics of his own terminal. Terminal characteristics revert to the system defaults when a dialup line is disconnected or when the user logs out.

For details of the software facilities associated with characteristics see the IAS/RSX-11D Device Handlers Reference Manual. For the setting of characteristics at system generation see the IAS System Generation Reference Manual.

FORMAT 4:

```
    PDS> SET
    FUNCTION?  TERMINAL
    ATTRIBUTE?  attribute
```

where:
attribute is either

terminaltype[DS]

or optionlist

where:

terminaltype	is one of ASR33, KSR33, ASR35, LA3ØS, LA3ØP, LA36, VTØ5, VT5Ø, VT52, VT61.
	SET TERMINAL terminaltype sets the characteristics other than the speed(s) to the default values listed in the <u>IAS/RSX-11D Device Handlers Reference Manual</u>, Table 2-3.
	If DS is appended, the speed also is set to the default value.
optionlist	is one or more of [NO]option option:value separated by spaces.

Each option and any short form listed with it may be abbreviated so long as it remains unique within the list of SET TERMINAL options. Each acceptable form of an option without a value may be negated by the prefix NO, e.g NOSCOPE.

An asterisk * marks the options which are likely to be most commonly used.

option can be:

	ALTMODE	is an old model Teletype which generates 175 or 176 (octal) when the ALT key is pressed. Either of these characters will be treated in the same way as ESCAPE.
	BACKSPACE	Terminal responds to the Backspace character
	BLOCKMODE	Terminal is a VT61 and is to be used in Block Mode
	CARRIAGERETURN or CR	Lines exceeding the terminal width as set are continued on the following line(s)
	COMPATIBLE	Terminal requires RSX-11M compatible Escape sequence handling.
	CONTROLCFLUSH or CCF	Flush type-ahead when CTRL/C is typed
*	DEFAULT	Restore terminal characteristics to system default values as existing at log-in time.

	ESCAPESEQUENCE	Terminal requires Escape Sequence recognition
*	FORMSMODE	Terminal is a VT61 and is to be used in Forms Mode.
	HANGUP	Hang up dialup line. This cannot be negated.
	HARDWAREFORMFEED or HFF	The characters Form Feed and Vertical Tab are recognized and do not need software simulation
	HARDWARETAB or HTAB	The character horizontal tab is recognized
*	HOLD	(VT5x and VT61 terminals only) used to enter auto-hold mode. Output from the computer will then be stopped automatically when the screen becomes full with output and may be resumed by pressing the SCROLL key to enable a further line to be output. Pressing the SHIFT and SCROLL keys simultaneously will enable a further page to be output. For this facility to work correctly the terminal must transmit and receive at the same speed.
	KEYBOARD	Terminal is capable of input.
	LOCALCOPY	Terminal echoes all characters as they are typed
*	LOWERCASEKEYBOARD or LCKEYBOARD	Lower case characters are accepted. If CTRL/R type-ahead is used, characters will be echoed as lower case, whether or not they are processed as lowercase. LOWERCASEKEYBOARD can be consistently used with NOLOWERCASEINPUT.
*	LOWERCASEINPUT or LCINPUT	Lower Case characters are to be passed to a program performing input even if program (e.g. EDI) asks for case conversion
LOWERCASEOUTPUT or LCOUTPUT or	LOWERCASEPRINTER or LCPRINTER	Terminal can print lower case characters
	LVF	LA36-type vertical fill is required for form feed and vertical tab (i.e. 66 nulls)
	NEWLINE	Terminal sends 'newline' when the carriage return key is pressed
	NONSTANDARDTAB or NSTAB	

		Terminal on receiving tab character does not space to the next 8-character boundary.
	NOPARITY	Do not generate parity bit on character output.
	PRINTER	Terminal is capable of Output.
	SCOPE	Terminal is a Scope (VDU) and rubout physically erases characters from the screen.
*	SIMULATEFORMFEED or SFF	Form feed and vertical tab are to be software simulated to start a new page and skip to next six-line boundary respectively.
*	TAPE	Terminal has a low speed Paper Tape Reader and interprets CTRL/B and CTRL/T accordingly. See Table 3-2.
	TWOSTOPBITS or TSB	Terminal requires two stop bits as normally required for mechanical printers e.g. ASR33.
	VERTICALFILL or VFILL	Terminal requires VTØ5-type vertical fill

option:value can be:

FILL:n	n is fill required for carriage return n = 7 supplies LA3ØS-type fill
LENGTH:n	n is page length in lines
NAME:name	name can be one of: ASR33, KSR33, ASR35, LA3ØS, LA3ØP, LA36, VTØ5, VT5Ø, VT52, VT55, VT61. This option is for use in 'deceiving' a program as to the type of terminal under which it is running, e.g. when mixed characteristics are required. The option sets only the location holding the name of the terminal type (IAS/RSX-11D Device Handlers Reference Manual, Table 2-3, column 1). To access this name from a program, see that same manual, Table 2-2, TC.TTP and Sections 2.4.3.5 through 2.4.3.7.

NOTE

SET TERMINAL NAME:name does NOT implicitly set the corresponding characteristics.

PARITY:type	type is EVEN or ODD. Set line to generate characters with parity. Note that parity is not checked on input.
READAHEAD:type	type is one of: NONE No read-ahead allowed DEFERREDPROCESSING or DP read-ahead accepted but not examined until a read which uses it is processed IMMEDIATEPROCESSING or IP read ahead is processed as it is typed but not echoed till it is read
SPEED:(m:n)	Set split-speed line. m is the keyboard (lower) speed. n is the printer or display (higher) speed
SPEED:n	Set line speed. n can be one of: speed in baud 134 (meaning 134.5 baud) EXTA (DH11 external speed A) EXTB (DH11 external speed B)
WIDTH:n	n is the page width in columns

EXAMPLES:

1. PDS> SET TERMINAL WIDTH:5Ø LENGTH:3Ø CR

   The width is set to 5Ø characters, the length to 3Ø lines. Lines of more than 5Ø characters are continued on the following lines.

2. PDS> SET TERMINAL

   ATTRIBUTE? SPEED:(15Ø:96ØØ)

   Terminal is to send at 15Ø baud and receive at 96ØØ baud.

3. PDS> SET

   FUNCTION? TERMINAL

   ATTRIBUTE? VTØ5 DS

   Terminal is a VTØ5 and is to run at the corresponding speed (24ØØ baud).

4. PDS> SET TERMINAL NAME:VT61

   The terminal type is recorded as being VT61 but no characteristics are thereby changed.

FORMAT 5 (SET NAMED LIST)

This format is available to users logged in under a UIC of [1,1].

FORMAT 5:

```
     PDS> SET

     FUNCTION?  TERMINAL:TTn
```

or

```
               TERMINAL:(TTm,...,TTn)

     ATTRIBUTE?  attribute
```

where:

m,...,n   are the unit numbers of the terminals to be affected.

attribute is as in FORMAT 4.

DESCRIPTION

This format sets the characteristics of terminal TTn or of terminals TTm,...,TTn to the values specified in attribute.

EXAMPLES

1. `PDS> SET TERMINAL:TT3 SPEED:(150:9600)`

   Terminal TT3 is to send at 150 baud and receive at 9600 baud.

2. `PDS> SET TERMINAL:(TT3,TT5,TT6) SPEED:300`

   Terminals TT3, TT5 and TT6 are to send and receive at 300 baud.

FORMAT 6

```
     PDS> SET PROTECTION [/OWN]

     FILE?  filespec

     PROTECTION?  (code)
```

or

```
     $SET PROTECTION [/OWN] filespec (code)
```

where

/OWN	if specified, changes the ownership UIC of the file to be the same as the UFD under which the file is stored.
filespec	is the specification of the file to which the protection code is to be applied.

(code) is the protection code to be applied to filespec. See Chapter 6, Section 6.1.3.

User categories are:

SYSTEM:

OWNER:

GROUP:

WORLD:

Types of access are:

R	read
W	write
E	extend
D	delete

Example

```
(SY:R, O:RWED, GRO:RW)
```

System has read access only. Owner has all four types of access. Group has read and write access only. World access remains unchanged.

EXAMPLES

1. PDS> SET PROTECTION/OWN CATHS.DAT

   PROTECTION? (GRO:R, SY:R, WORLD:, O:RWDE)

2. $SET PROTECTION TONY.MAC (OW:RWED, SY:, GR:, W:)

3. PDS> SET PRO MYPROG.COB (SY:RWED, OW:RWDE, WO:DERW, GR:RWED)

FORMAT 7 (SET PRIORITY)

The SET PRIORITY command the user to alter the priority of an active task.

FORMAT

```
PDS> SET PRIORITY
TASK? taskname [terminal] priority
```

where

taskname	is the installed name of the task being altered
terminal	is the terminal from which the task to be altered was activated. The default is the current terminal.
priority	is the new task priority (that is, a decimal number ranging from 1 to 25Ø)

EXAMPLES

1. PDS> SET PRIORITY SCAN TT4 12Ø

   Sets the priority of the installed task SCAN running from terminal 4 to 12Ø.

2. PDS> SET PRIORITY XYZ,,13Ø

   Sets the priority of the installed task XYZ to 14Ø. XYZ was invoked from this terminal.

SHOW

The SHOW command causes the terminal to display specified information at the user's terminal. The parameter to SHOW determines the type of information displayed.

FORMAT

```
     PDS> SHOW

     ATTRIBUTE? parameter
```

where

parameter specifies the type of information to be displayed. The options are:

CLI	Display information about the Command Language Interpreters (CLIs) currently running in the system.
DEFAULT	Display the user's current default device and UFD
DEVICES	Display information about all or selected devices known to the system. See the section called Devices below. With /PUD, displays also the PUD address of the device unit(s).
DAYTIME	Display the current time and date.
MEMORY	Display the use of the system's memory.
STATUS	Display information about the current status of the user's job.
GLOBAL_AREAS	Display information about resident global areas.
LUNS	Display current assignment of luns for an installed task. PDS prompts for the task name.

Devices

The command SHOW DEVICES causes the system to display at the user's terminal the symbolic names of the devices known to the system. The user can choose to print information about one particular device (e.g. DKØ), all devices of that type (e.g. DK) or all devices. The Physical Unit Directory (PUD) addresses of the units can also be requested. Physical device names are followed by "**" if they are currently available for use. System logical device names are followed by the associated physical device names. The listing also includes messages giving additional information about particular devices. The messages and their meanings are:

Message	Meaning
GLOBAL	The device has been mounted globally (see the MOUNT command).
MOUNTED	The device is mounted.
REALTIME	The device is mounted for realtime activity.
T/S DEVICE	The device is a timesharing device. If followed by an 'X' (see example) the device has been explicitly allocated to a user.
T/S TERMINAL	The terminal is a timesharing device.
SYSTEM	The device is a system device.
SPOOLED:n	The device is spooled. n is the current setting of the forms type.
TIMESHARING:n	n is the number of timesharing users accessing the device.

Memory

The command PDS> SHOW MEMORY displays on a VDU terminal (VT05, VT50, VT52, VT55, VT61) the memory useage and task activity of the system provided that the terminal handler was configured to support escape sequences.

The display appears in two rows of columns (one row only on a VT50). Each column refers to a portion of memory.

All types of task area within the occupied memory are displayed by task name. Shareable global areas are displayed by name.

Fixed tasks are displayed as FIXED until the task becomes active. Tasks listed down the right hand side of the screen are on the Memory Required List (MRL). The number of nodes available and the largest hole are included in the heading information at the top of the screen. The name of the currently active task, and the terminal for which it is running, are also displayed only if the SHOW MEMORY task (...DEM) is run as a high priority real time task.

On the display, at the bottom of each column,

```
<->  indicates a task's read/write (impure) area
<=>  indicates a task's read/only (pure) area
[=]  indicates a shared global area (SGA)
<+>  indicates a fixed or non-checkpointable task
```

Once the memory diagram is displayed, the portion of memory being displayed can be altered dynamically by one of the following commands. Do NOT type CTRL/C or use the control key with these commands.

FORMAT (no prompt):

```
     B[ASE] base

where

               base is the beginning  of  the  area  of  memory  whose
               activity  is to be displayed.  'base' is entered either
               in the form

               mK   that is, mK words (m decimal)

               or in the form

               n    that is, n octal blocks of 32 words or 100 (octal)
                    bytes

     G[RAIN] grain
               Reset the amount of memory  referred  to  by  a  single
               column  of  the  display.  grain has the same syntax as
               base.

     C[LEAR]   Clear the VDU screen and redisplay.  Used, for example,
               to clear an external message from the screen.

     E[XTENT] nK
               Change extent of display

     E[XTENT] ALL
               Display all memory (initial state)

     I[NTERVAL] n
               Update display every n seconds (initially n = 1)

               n = 0 gives continuous update.

     X         Exit to PDS.
```

EXAMPLES

1.
```
PDS> SHOW DAYTIME

10:53:41   1-JUN-77
```

2.
```
PDS> SHO DEV
TT0      **       T/S TERMINAL
CI0      TT0
CO0      TT0
CL0      LP0
TO0      TT6
SP0      SY0
PI0      **
MO0      **
MM0      **       T/S DEVICE X  MOUNTED                 TIMESHARING:1
DT1      **       T/S DEVICE
DT0      **       T/S DEVICE
LP0      **       SYSTEM        SPOOLED:0               TIMESHARING:8
TT11     **       T/S TERMINAL
TT10     **       T/S TERMINAL
TT7      **       T/S TERMINAL
TT6      **       T/S TERMINAL
TT5      **       T/S TERMINAL
TT4      **       T/S TERMINAL
TT3      **       T/S TERMINAL
TT2      **       T/S TERMINAL
TT1      **       T/S TERMINAL
DS0      **                     MOUNTED     GLOBAL
DB1      **       T/S DEVICE X                          TIMESHARING:1
DB0      **       SYSTEM        MOUNTED     GLOBAL TIMESHARING:8
DK1      **
DK0      **       T/S DEVICE
SY0      DB0
```

3.
```
PDS> SHOW DEV/PUD DB

DB1  152404  **  T/S DEVICE X                         TIMESHARING:1
DB0  152470  **  SYSTEM        MOUNTED     GLOBAL TIMESHARING:8
```

SORT

The SORT command is used to sort files into a specified sequence. Consult the PDP-11 SORT Reference Manual before using this command.

FORMAT

```
     PDS> SORT/qualifiers1

     FILE?  in-filespec/qualifiers2
```

where

qualifiers1 are any of the following:

/OUTPUT:outfilespec
: Specifies the output file. The default filetype is DAT. If /OUTPUT is omitted and the in-filespec contains no version number, the output file is assumed to be the same as the input file, with the version number incremented. If /OUTPUT is omitted but a full in-filespec is given, the output filespec is assumed to be exactly the same as the input filespec (that is, the version number not incremented).

/ALLOCATION:n
: Specifies the initial space allocation for the output file before the sort process begins. n is the number (decimal) of bytes.

/BLOCK_SIZE:n
: For magtape files only, specifies a non-standard tape block size. n is the number (decimal) of bytes.

/BUCKET_SIZE:n
: Specifies the RMS bucket size of the output file. n is the number (decimal) of bytes.

/CONTIGUOUS
: Specifies that the initial space allocation for the output file to be contiguous.

/DEVICE:device or
/DEVICE:(/qualifiers) or
/DEVICE:(device:/qualifiers)
: For special applications requiring control of the SORT scratch files, specifies the scratch file device.

  device is the scratch file device.

  /qualifiers can be one or both of

      /ALLOCATION:n
      /CONTIGUOUS

/FILES:n
: For special applications, specifies the number of scratch files to be used by SORT (n must be between 3 and 8).

/FORMAT:format[:n]

Specifies the record format of the output file.

format is one of

FIXED
VARIABLE
UNKNOWN

[n] is optional and specifies:

record length (with FIXED)

maximum record length (with VARIABLE or UNKNOWN).

/KEYS:(abm.n:......)

Defines the key fields to SORT where:

'a' defines how to treat the data (i.e. character, zone, etc). The default is character.

'b' is the general sort order, where:
'N' is normal (ascending)
'O' is opposite
The default is 'N'.

'm' is the first position of key field. This must be defined.

'n' is the length of key field. This must be defined.

A maximum of 10 keys can be specified. The major key is the first in the string, and the minor key is the last.

/PROCESS:x — Defines the type of SORT process, where x is one of

RECORD (default)
TAG
ADDRESS_ROUTING
INDEX

/SEQUENTIAL or
/RELATIVE — Specifies the file organization of the output file.

/SPECIFICATION:file-spec
The control parameters for SORT are contained in the specified file. The default filetype is SRT.

in-filespec — is the filespecification of the file to be sorted.

qualifiers2 — is the first or both of the following.

/FORMAT:format:n
is mandatory and specifies the record format and record length of the input file.

format is one of
FIXED
VARIABLE
UNKNOWN

n specifies record length for FIXED length records and the maximum length of VARIABLE or UNKNOWN structured records.

/INDEXED:n — is mandatory for an input file with Indexed Sequential organization, where n is the number of keys.

See the PDP-11 SORT Reference Manual for further details.

EXAMPLES

1. <u>PDS></u> SORT/KEY:C1.4

   <u>FILE?</u> CAROL.DAT/FORMAT:UNKNOWN:13Ø

   Sort the file CAROL.DAT according to the characters in the key. The key is in position 1 of the record and is 4 bytes long. Name the output (sorted) file as CAROL.DAT with incremented version number.

2. <u>PDS></u> SORT/SPEC:FRANK.SRT

   <u>FILE?</u> MARTIN.DAT;3/FOR:FIXED:124/INDEXED:5

   Sort the file MARTIN.DAT;3 according to the specifications held in FRANK.SRT. Name the output (sorted) file as MARTIN.DAT;3, to replace the input file.

3. <u>PDS></u> SORT/KEYS:(BN1.6 C8.2)/REL

   <u>FILE?</u> TELEPHONE.LST/FORMAT:FIXED:4Ø

4. <u>PDS></u> SORT/SPEC:STOCK.CTL/DEV:(/ALL:1ØØ/CO)

   <u>FILE?</u> P127Ø9.ØØ1/FORMAT:VAR:8Ø

STOP

The STOP command is used only in an indirect command file or a batch command file. STOP prevents all further processing within the file.

FORMAT

```
[$]STOP
```

DESCRIPTION

STOP can be used by itself or as the action in an ON command.

EXAMPLE

```
$JOB DEMO
$ON WARNING GOTO L1Ø
$RUN JOB1
$GOTO L2Ø
$L1Ø:  RUN TEST
$STOP
$L2Ø:  ON WARNING STOP
$RUN JOB2
$RUN JOB3
$EOJ
```

SUBMIT

The SUBMIT command sends a file containing batch commands to the batch processor.

FORMAT

```
     PDS> SUBMIT [/PRIORITY:n] [/NOTRANSFER]

     FILE? filespec
```

where

n	is the priority at which the file is to be submitted (for example, priority 1). n must be between 1 and 1ØØ.
/NOTRANSFER	inhibits the copying of filespec to SP.
filespec	is the specification of a file containing batch commands. The specification must contain a filename. The default filetype is .BIS.

DESCRIPTION

The system submits the filename of the file of batch commands filespec to a queue of jobs for subsequent processing in batch mode.

Unless filespec exists on a system device (that is, available to all timesharing users) or unless /NOTRANSFER is specified, filespec is automatically copied to device SP. SUBMIT/NOTRANSFER can be used when the device on which filespec exists will still be mounted when the job is dequeued.

EXAMPLES

```
    1.   PDS> SUBMIT

         FILE? BATCHFILE.BIS

    2.   PDS> SUBMIT/PRIORITY:6
         FILE? BATCHJOB

    3.   PDS> SUBMIT/NOTRANSFER DK1:MYJOB

    4.   $SUBMIT MYJOB
```

TYPE

The TYPE command causes the contents of one or more specified files to be printed at the user's terminal. In batch, the file is output directly to the batch log.

FORMAT

```
     PDS> TYPE

     FILE? filespec-1[,...filespec-n]
```

where

filespec	is the specification of a file. The specification must contain a filename and filetype. Wild-cards are allowed.

EXAMPLES

```
     1.   PDS> TYPE

          FILE? (BARLEY.CBL;2,  GRAHAM.CBL;2)

     2.   PDS> TYPE APPLE.DAT

     3.   $TYPE MKEE6.CBL
```

UNFIX

The UNFIX command allows the user with PDS Command Privilege PR.RTC to free a fixed task from memory.

FORMAT

```
     PDS> UNFIX
     TASK?  taskname
     [TERMINAL?  terminal]
```

where

taskname	is the installed name of the task to be unfixed from memory.
terminal	is the terminal at which the task is to be unfixed. This default is the current user's terminal.

EXAMPLES

```
     PDS> UNFIX JK03

     PDS> UNF MKEE9 TT6
```

UNLOCK

The UNLOCK command unlocks a file that was locked as a result of being improperly closed.

FORMAT

```
     PDS> UNLOCK

     FILE?  filespec-1[,...,filespec-n]
```

or

```
     $UNLOCK filespec-1[,...,filespec-n]
```

where

filespec	is the specification of the file to be unlocked. Wild-cards are allowed.

DESCRIPTION

If a program using File Control Services (FCS) has a file open with write access and exits without first closing the file, the file will be locked against further access as a warning that it may not contain proper information. Typically the following information would not have been written to the file:

1. The current block buffer being altered.

2. The record attributes which contain the end of file information.

   By using the UNLOCK command, the user can access the file and can determine the extent of the damage and perhaps take appropriate corrective action.

EXAMPLE

```
     PDS> UNLOCK
     FILE? THAMES.MAC;7
```

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under SPR service, submit your comments on an SPR form.

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name ____________ Date ____________

Organization ____________

Street ____________

City ____________ State ____________ Zip Code or Country ____________

Please cut along this line.

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

```
Software Documentation
146 Main Street  ML5-5/E39
Maynard, Massachusetts  01754
```

digital

digital equipment corporation

Printed in U.S.A.