pdp 11

RSX-11M
Beginner's Guide

Order No. DEC-11-OMBGA-A-D

digital

RSX-11M
Beginner's Guide

Order No. DEC-11-OMBGA-A-D

June 1977

This document describes how to use the RSX-11M operating system
from a terminal to develop a simple FORTRAN program and to
manipulate files.

digital equipment corporation · maynard, massachusetts

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM	DECSYSTEM-20	TYPESET-11

CONTENTS

PREFACE

MANUAL OBJECTIVES AND READER ASSUMPTIONS

The information presented in this guide enables a new user of RSX-11M to perform simple program development. The guide assumes a knowledge of computing in general, but it assumes no knowledge of the RSX-11M operating system. RSX-11M terms are defined when they appear for the first time.

STRUCTURE OF THE DOCUMENT

The guide has been structured so that the new user can learn terms and procedures as he or she actually creates and runs a program. The program development procedures are divided into three parts, corresponding to the following three chapters:

- THE TERMINAL
  Chapter 1 discusses keyboard facilities, how to log onto the terminal, and how to issue commands to the operating system.
- PROGRAM PREPARATION
  Chapter 2 describes how to use system programs to create, compile, link, and run a simple program.
- THE FILES
  Chapter 3 describes how to specify and manipulate files.

The development of a simple FORTRAN program that adds two numbers together is used as an example throughout the guide. The Appendix reproduces the development of that program without the explanations provided in the previous chapters.

ASSOCIATED DOCUMENTS

The guide makes frequent references to other RSX-11M documents, which are all described in the *RSX-11M/RSX-11S Documentation Directory*. The directory defines the intended readership of each manual in the RSX-11M/RSX-11S set and provides a brief synopsis of each manual's contents.

DOCUMENTATION CONVENTIONS

The following notes describe documentation conventions used in this guide.

- Text in italics, other than document titles or section headings, indicates a new RSX-11M term.
- Brackets ([ ]) indicate optional elements of command input (ABO[RT]), except when the brackets are a part of a User Identification Code ([300,221]).

- In text that illustrates command format, upper-case letters denote actual command input. Lower-case letters indicate parameters that must be defined by the user. For example:

  ABO taskname

  The user types the letters ABO but must replace taskname with the name of an actual program.
- In examples of terminal dialogues, text in a contrasting color indicates user input, as opposed to computer-generated text shown in black.

CHAPTER 1

THE TERMINAL

A terminal provides the means of communication between you and the RSX-11M operating system. You instruct the system by typing commands at the terminal; the system either responds as instructed or displays an error message telling you that it does not understand or cannot respond to your input.

RSX-11M supports a variety of terminals, which fall into two general categories: hard-copy terminals that print on paper, and Cathode Ray Tube (CRT) terminals that display characters on a screen. Most CRT terminals cannot provide a permanent record of terminal activity. Figure 1-1 illustrates a typical hard-copy terminal (an LA30 terminal in this photograph). Figure 1-2 at the top of the next page illustrates a typical CRT terminal (a VT50 terminal).

Figure 1-1   An LA30 Hard-Copy Terminal
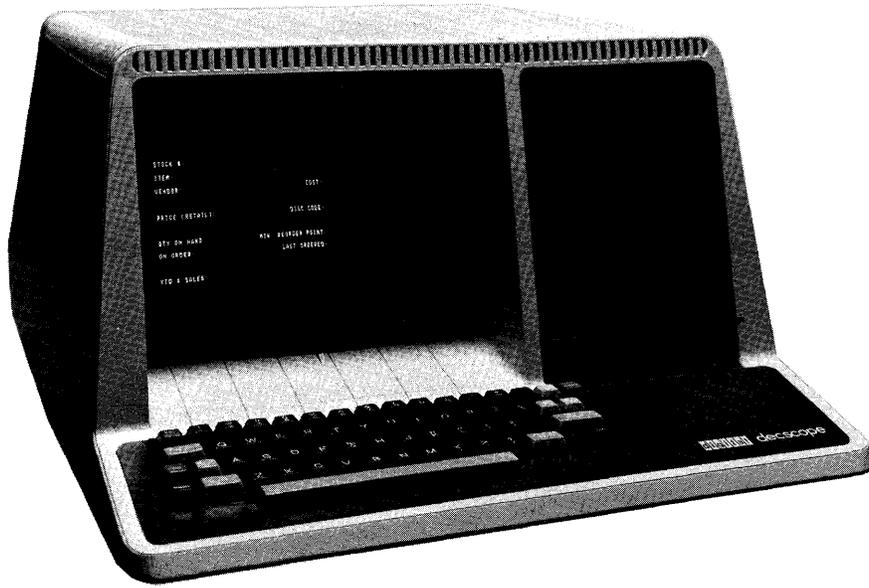
Figure 1-2   A VT50 CRT Terminal

1.1   USING A TERMINAL

All terminals have a keyboard that has the same basic configuration as a typewriter. In addition, the keyboard includes other special keys that enable you to execute various computer functions. Individual keys are not always in the same position on different types of terminals, so check the keyboard layout each time you use a different terminal for the first time. Figure 1-3 shows the keyboard of an LA30 terminal. Figure 1-4 on the next page shows the keyboard of a VT50 terminal, which has a keyboard layout different from that of the LA30 terminal.
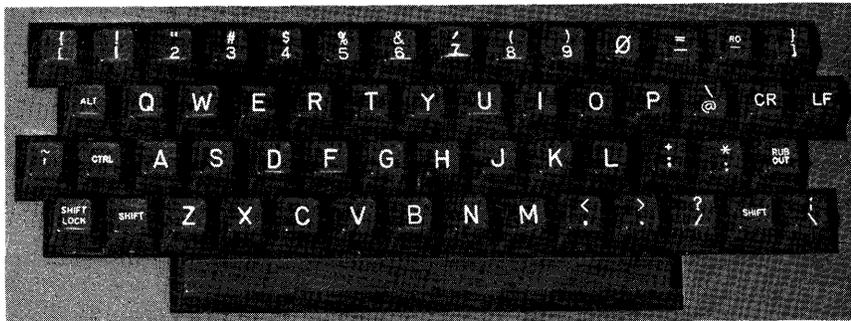
Figure 1-3   Keyboard of an LA30 Terminal
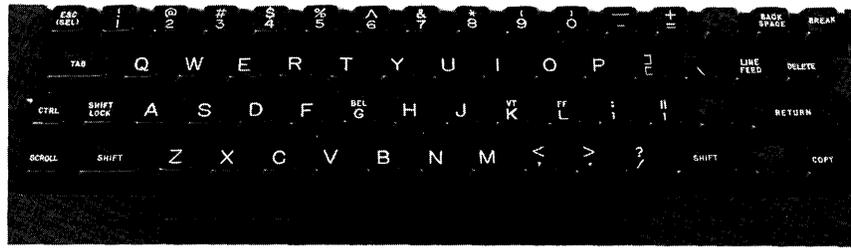
Figure 1-4   Keyboard of a VT50 Terminal

1.1.1   Function Keys

Important function keys that appear on the terminals are described in Table 1-1.

Table 1-1   Function Keys

Function Key	Description
CR or RETURN	Terminates an input line and advances the carriage or cursor to position 1 of the next line.
CTRL	Is part of numerous 2-key combinations (CTRL and a letter key) that perform a variety of functions. Each valid combination is called a control character, represented in this manual by CTRL/X, where X is the variable letter. The control characters you need to use most often are described in Section 1.1.2.
RUBOUT or DEL	Deletes the last character typed on the current line, and contiguous characters to the left if you press the key repeatedly. Some CRT terminals erase each deleted character from the screen and move the printing position back one space. Otherwise, the terminal prints a backslash (\), then each deleted character, then another backslash before it prints the first correct character (for example, MISTKAE\EAK\AKE).
TAB	Advances the printing position to the next tab stop. Tab stops occur after every eighth character position in the line.

1.1.2   Control Characters

A *control character* is entered by striking the appropriate letter while pressing CTRL. The system responds to some control characters by displaying an up arrow (↑) followed by the appropriate letter (↑U, for

example). Other control characters are not echoed in a display. See Section 2.1.1 in the *RSX-11M Operator's Procedures Manual* for a complete list of the control characters. The list in Table 1-2 describes those most commonly used.

Table 1-2 Control Characters

Character	Description
CTRL/C	Gains the attention of the Monitor Console Routine (MCR), which interprets commands to the operating system. (See Sections 2.1.1 and 2.1.3.3 in the *RSX-11M Operator's Procedures Manual* for more information about the use of CTRL/C.) In most cases the system responds to CTRL/C by displaying the prompt MCR>
CTRL/O	Alternately discards and resumes the display of output sent to the terminal. Discarded output is lost. For example, if you are running a program that generates unwanted output, type CTRL/O. The system then discards that program's output until you type CTRL/O again. (Note that the only program affected is the one sending output to your terminal when you type CTRL/O.) See the *RSX-11M Operator's Procedures Manual* for a more detailed description of the CTRL/O function.
CTRL/Q and CTRL/S	CTRL/S delays the display of further output until you type CTRL/Q to restart it. For example, if you are using a CRT terminal that displays output too quickly to be read easily, press CTRL/S to halt the display; then, when you have read the screen, press CTRL/Q to restart the output display. Repeat the process as necessary.
CTRL/R	Performs a carriage return and retypes the current line, omitting any deleted characters. If the DELETE or RUBOUT key on your terminal prints backslashes and deleted characters, the current line may be difficult to read because it contains

Table 1-2 (Cont.) Control Characters

Character	Description
	numerous extraneous characters. Before terminating the line, type CTRL/R to ensure that you have made the right corrections. For example: MISTKAE\EAK\AKE <CTRL/R> MISTAKE
CTRL/U	Deletes the current line and performs a carriage return. This function allows you to retype a line when individual deletions would be impractical. Remember to type CTRL/U before you terminate the incorrect line.
CTRL/Z	Used by many tasks as a signal to exit and return control to MCR.

1.2 MCR COMMANDS

The commands that control system operation from a terminal are called *MCR commands*. MCR stands for the Monitor Console Routine, which interprets terminal input. You communicate with MCR by entering a command string in the following format:

command-name parameter(s)/keyword(s) line-terminator

where:

command-name   consists of 3 or more letters, terminated by a space, that uniquely identify an MCR function. MCR reads the first 3 letters only; further letters merely help you to identify the command. Some examples of command names are given below. (Brackets enclose optional input.)

Names	Function
ABO[RT]	ABOrt a running program (that is, stop it from running).
DMO[UNT]	DisMOunt a volume.
UFD	Create a User File Directory (see Section 3.1.2).

parameter usually specifies the object of the command function, which is most often a task or a device. (A task is an executable program.) One or more blank spaces must separate the parameter from the command name, and one parameter from another. For example, when you issue the Abort (ABO) command, you include the name of the running task to be aborted.

- ABO TASK
  Abort the task named TASK.
- RUN TASK 12:35:00
  Run the executable program called TASK at 12:35:00.

/keyword modifies either a parameter or the function of the command. A keyword consists of a slash (/) and an alphanumeric string indentification, optionally followed by an equal sign (=) and a keyword value:

/keyword[=value]

A keyword immediately follows the parameter it defines:

parameter/keyword[=value]

However, when a keyword modifies a command function, you must insert at least one space between the name and the keyword:

command-name /keyword[=value]

Spaces are not required between consecutive keywords:

parameter/keyword/keyword. . .
command-name /keyword/keyword. . .

Examples:

- SET /CRT=TI:
  Enable backspace deletion on your CRT terminal. TI: always stands for the terminal you are currently using. The keyword /CRT determines the function of the SET command.
- ABO TASK/PMD
  Abort the task named TASK and produce a post-mortem dump, which is a listing that describes the state of the task when it was aborted.

line-terminator is either the CR or RETURN key, which performs a carriage return (<CR>), or the ALTmode or ESCape key (<ESC>). Because <ESC> has special significance in certain circumstances, this manual uses <CR> as a line terminator.

When you press <CR>, the system sends the current line to MCR.

1.2.1 Preparing to Issue Commands

Before typing a command, you must ensure that the terminal is in the right state to receive input directed to MCR:

- Check that the terminal's power is on.
- Check that the LOCAL/REMOTE switch is set to REMOTE.
- Consult installation instructions for additional required terminal settings and dial-up instructions.
- Press the CR or RETURN key to clear the terminal and to obtain the default prompt (>); or type CTRL/C to obtain the explicit MCR prompt (MCR>).

Both the explicit MCR prompt and the default prompt indicate that MCR can receive input typed at the terminal. Any other prompt indicates that the terminal will send the input to a task other than MCR. Both DIGITAL-supplied and user-written tasks can request input by displaying a task prompt at a terminal. If you see a task prompt, someone else is probably using that terminal.

Examples in this manual use the default prompt, which a terminal displays most often when it is ready to receive input to MCR.

NOTE
If you are working with a system that does not support multiuser protection, you do not need to log on and off the terminal. Ignore the next two sections (1.2.2 and 1.2.3) and proceed directly to Section 1.3.

1.2.2 Logging On and Off a Terminal

To begin a session at the terminal, you must issue the Hello command to log on. The logging-on procedure has several purposes:

- It allows the system to ensure that you are an authorized user and to record information about your usage of the system.
- It grants you the use of the terminal for access to the system until you log off.
- It establishes initial defaults for your terminal activities.

The Hello Command

Hello is an interactive command that prompts for its parameters, as follows:

>HELLO <CR>

ACCOUNT OR NAME: user-id-code
                 last-name   } <CR>

PASSWORD: password

where:

user-id-code	is a code, commonly called a UIC, that basically consists of two octal numbers that identify your account on the system. The first number (g) stands for your user group; the second number (m) is your own number within the group. The general UIC format encloses the two numbers, separated by a comma, in brackets: [g,m] .
last-name	is your last name, which you can enter instead of a UIC. The system derives the correct UIC from the name you specify.
password	is a 1- to 6-character alphanumeric string. The account information maintained by the system includes the correct passwords for each UIC and last name. You cannot gain access to the system unless you type the password that corresponds to the UIC or name you have entered.
	So that your password remains private, the system does not print the characters you type in response to PASSWORD:.

If you do not know your UIC or password, contact the system manager, or whoever controls the use of the system at your installation.

Example:

```
>HEL <CR>
ACCOUNT OR NAME: CHARLES <CR>
PASSWORD: GREY <CR>

          RSX-11M BL18 MULTI-USER SYSTEM

GOOD MORNING
22-MAY-77 11:07 LOGGED ON TERMINAL TT4:

22-MAY-77

     SYSTEM WILL BE DOWN TODAY FROM 13:00-15:00 FOR
     CORRECTIVE MAINTENANCE

>
```

When the system receives the correct password, the terminal displays a logon greeting that includes a system identification and the date and time you logged onto the terminal. In addition, the terminal optionally displays further messages, which usually supply information that affects general use of the system. The default prompt (>) on a new line after the messages indicates that you can proceed to issue further MCR commands.

If the message

```
HEL -- OTHER USER LOGGED ON
```

appears when you issue the Hello command, someone else is using the terminal. Either try to log onto a different terminal or seek out the person who logged on before you. Do not proceed to use the terminal yourself without the other user's knowledge.

The Help Command
Help is the only MCR command that you can issue before logging on. Because the first 3 letters are the same as Hello, you must enter HELP in full. When you issue this command, the terminal displays text (determined by each installation) that generally tells you how to log on and how to issue further commands. Even if you do not need help initially, issue the command at least once so that you know the nature of the information it provides.

The Bye Command
Whenever you finish a session, you must log off in order to make your terminal available to other users. Log off by typing

>BYE <CR>

This command has no parameters. The system responds by performing all actions required to terminate your activity at the terminal. (See the *RSX-11M Operator's Procedures Manual* for more details.) A message like the following then appears:

```
>
HAVE A GOOD MORNING
22-MAY-77 11:34 TT4:   LOGGED OFF
>
```

You must reissue the Hello command to log on if you want to issue further commands other than Help.

1.2.3 The Quickest Ways to Log On
Most users want to type as little as possible when using a terminal; they also want the quickest possible response time. This section describes ways to shorten the logging-on procedure.

Suppressing Optional System Messages

Because the optional system messages that follow the Hello command greeting can sometimes be lengthy and usually need to be read only once a day, the Hello command allows you to enter a special form of the UIC parameter to suppress the display of these messages. The normal format for a UIC is [g,m]. However, to signal that you do not want to see the optional system messages, replace the comma with a slash (/), as follows:

[g/m]

In all, Hello permits four UIC representations:

g/m
[g/m]
g,m
[g,m]

Either slash format always suppresses the optional messages. Brackets around the UIC are optional. Note, however, that a UIC must always be specified in the format [g,m] when it is not a parameter to the Hello command.

If you have provided your name or the comma format of UIC, you can stop the message display by typing CTRL/O. The system then immediately displays the default prompt. For example:

```
> HEL <CR>
ACCOUNT OR NAME: CHARLES   <CR>
PASSWORD:               <CR>

          RSX-11M BL18 MULTI-USER SYSTEM

GOOD MORNING
22-MAY-77 12:14 LOGGED ON TERMINAL TT4:

22-MAY-77

     SYSTEM WILL BE DOW<CTRL/O>

>
```

Suppressing ACCOUNT OR NAME: Prompt

Another way to shorten the time it takes to log on is to enter the Hello command and UIC (or last name) on the same line:

```
> HEL 201/312 <CR>
PASSWORD:
```

Entering the command in this way eliminates the need for the prompt ACCOUNT OR NAME:.

1.3 ERROR MESSAGES

When MCR receives input that it does not recognize or knows to be incorrect or invalid, it displays an *error message*. An example is the message that appears if you try to log onto a terminal already being used by someone else (see *The Hello Command* in Section 1.2.2). Another message related to the Hello command can be returned if you enter a UIC (or last name) or password that is not stored in the system's account file:

```
>HEL <CR>
ACCOUNT OR NAME: PEEK <CR>
PASSWORD: JOHN <CR>
HEL ---- INVALID ACCOUNT
>
```

This attempt to log on is not successful either because the system does not recognize the last name (PEEK), or because JOHN is not the correct password.

How you should react to an error condition depends on the message displayed. All error messages returned by MCR commands and MCR-related tasks are explained in the *RSX-11M Operator's Procedures Manual*. Individual command descriptions given in that manual include a list of possible error messages; Chapter 7 of that manual details all MCR error messages in alphabetical order. (Each error message displayed at your terminal begins with the 3-letter name of the associated command or task.) When you encounter an error while running a system task, such as a text editor, look for clarification in documentation that describes that software.

CHAPTER 2
PROGRAM PREPARATION

The steps required to prepare a program to run on RSX-11M are as follows:

1. Create a source program.
2. Compile or assemble the source program to produce object code.
3. Link the object code to create an executable program, called a *task*.
4. Issue an MCR command to run the task.

Figure 2-1 illustrates this procedure in a diagram. Each of the steps involves the manipulation of a *file*, which is an owner-named area on a magnetic medium. This magnetic medium, called a *volume*, is used to store data. Disks, DECtapes, and ANS magnetic tapes[1] are all volumes.
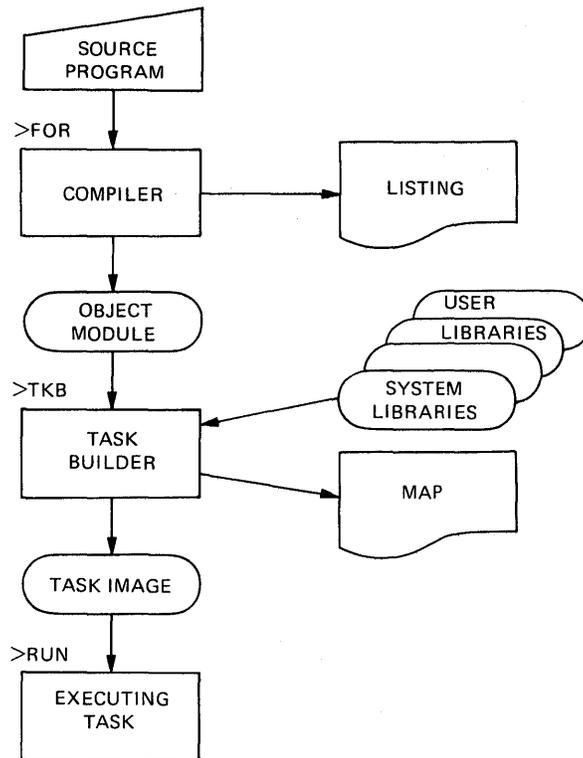
Figure 2-1   Steps in Creating a FORTRAN Task

---

[1]ANS magnetic tapes are tapes that conform to the American National Standard for Magnetic Tape Labels for Information Interchange, X3.27-1969.

When you log onto a terminal, the system automatically allows you access to a disk; this disk is your *default system disk*, on which, unless you specify otherwise, the system stores all your files. (If your system does not support multiuser protection, you have automatic access to a system disk that is accessible to all users.) The following sections on program preparation assume you are using your default system disk.

2.1 THE TEXT EDITOR (EDI)

The RSX-11M Text Editor (EDI) is the system program used to create a source program file.

2.1.1 Creating a Source Program File

To invoke EDI, issue a call to the editor in the same way that you issue an MCR command:

>EDI <CR>

The editor responds by displaying its task prompt:

EDI>

To create a file, you then specify a *file name* and a *file type* that describe the intended file contents in the following format:

EDI>filename.type

where:

filename   is a 1- to 9-character alphanumeric string.

type   is a 3-letter mnemonic, preceded by a period (.), related
            to the file contents. The following types are standard
            for programming language source files:

Type	Language
BAS	BASIC
CBL	COBOL
FTN	FORTRAN
MAC	MACRO

To illustrate the creation of a source file, the following examples deal with a FORTRAN program called ADD.FTN, an interactive program that adds two numbers together and displays the result.

```
>EDI <CR>
EDI> ADD.FTN <CR>
[CREATING NEW FILE]
INPUT
```

When EDI receives the name of a file that does not exist, it creates an empty file with the given name and displays the last two lines shown in the example above. You can then immediately start to type in your source program. When you terminate each line of input, the system stores it in a buffer, which EDI subsequently writes to the new file.

Use the keyboard facilities described in Tables 1-1 and 1-2 to correct any input mistakes on the current line. Once a line has been terminated and written to the file, you must use editor commands to correct it.

The following example specifies the new file on the same line as the call to EDI; this format shows an alternate (and quicker) way to summon the editor to create a file (Note the use of the DELETE or RUBOUT key, CTRL/R, and CTRL/U).

```
>EDI ADD.FTN <CR>
[CREATING NEW FILE]
INPUT
        TYPE 1 <CR>
1       FORMAT (' ENTER TWO NUMBERS - M,N') <CR>
        APPE\EPP\CCEPT 2,K,L <CTRL/R>
        ACCEPT 2,K,L <CR>
2       FORMAT (22\2\I5) <CR>
        PRINT ^U
        TYPE 3,K+L <CR>
3       FORMAT (' THE SUM IS ',I5) <CR>
        STOP <CR>
        END <CR>
<CR>
*EX <CR>
[EXIT]
>
```

Note the last four lines of the example above:

```
<CR>
*EX <CR>
[EXIT]
>
```

After terminating the last line in the program, type <CR> as the first character in the new line. EDI responds by displaying an asterisk (*) prompt. Until this point, EDI has been operating in *input mode*, entered automatically when EDI created the new file. Typing carriage return (<CR>) at the beginning of a new line switches EDI from input mode to *edit mode*; the asterisk is a prompt for *editor commands*. The command EX instructs EDI to close the file ADD.FTN and then to exit to MCR.

2.1.2 Displaying a Copy of the New File
To display a copy of the new file, issue the command

```
>PIP TI:=ADD.FTN <CR>
```

This command requests the Peripheral Interchange Program (PIP) to display a copy of ADD.FTN at your terminal. TI: (Terminal Input) is a device name that always represents the issuing terminal. (See Chapter 3 for more information about PIP.) When entering this PIP command, remember to type an equal sign (=) between TI: and the name of the file to be copied:

>PIP TI:=filename.type <CR>

Example:

```
>PIP TI:=ADD.FTN <CR>

        TYPE 1
1       FORMAT (' ENTER TWO NUMBERS - M,N')
        ACCEPT 2,K+L
2       FORMAT (2I5)
        TYPE 3,K+L
3       FORMAT (' THE SUM IS ',I5)
        STOP
        END
>
```

2.1.3 Editing a Source Program File

To edit an existing file, enter the EDI command:

```
>EDI ADD.FTN <CR>
```

Note that this is the same command line entered to create the file ADD.FTN. However, because ADD.FTN now exists, EDI responds differently:

```
>EDI ADD.FTN <CR>
[00009 LINES READ IN]
[PAGE 1]
*
```

EDI retrieves the existing file and automatically enters edit mode, indicated by the messages and the asterisk prompt. The message [00009 LINES READ IN] tells you the number of lines from the input file that EDI has read into a buffer. The lines within the buffer comprise the current block of text to be edited (initially [PAGE1]). The buffer may or may not contain the complete input file depending on the sizes of the file and the buffer. To access text beyond the current block, you issue an editor command (Renew, see Table 2-1) that writes the current block to the output file and refills the buffer with the next block of text.

An internal *line pointer* determines the line within the block to be edited. When EDI reads in a buffer, the line pointer points to a line immediately preceding the first line of text, thus allowing you to insert one or more lines at the top. You subsequently position the pointer by searching for a particular piece of text or by using commands that reposition the pointer.

Locating and Changing Text

Most EDI Commands can be abbreviated to one or more letters. In the following text, the optional portion of each command is underlined.

The example below illustrates the following editor commands:

Command	Function
<u>LO</u>CATE	Locate a string of text in the current block.
<u>C</u>HANGE	Replace one text string with another.
<u>N</u>EXT	Advance the line pointer to the next line.
<u>P</u>RINT	Print the current line.
<u>TO</u>P	Position the line pointer at the top of the current block.
<CR>	Point to and print the next line.

Example:

The text following each break in this example explains the command that has just been given to the editor.

```
>EDI ADD.FTN <CR>
[00009 LINES READ IN]
[PAGE 1]
*LOCATE ENTER <CR>
```

Point to and print the line containing the word ENTER.

```
1        FORMAT (' ENTER TWO NUMBERS - M,N')
*CHANGE/ENTER/TYPE/ <CR>
```

Change the word ENTER to TYPE. (Note how the slashes delimit both text strings.) EDI then prints the corrected line.

```
1        FORMAT (' TYPE TWO NUMBERS - M,N')
*NEXT <CR>
```

Point to the next line.

```
*PRINT <CR>
```

Print the current line.

```
        ACCEPT 2,K+L
*LOCATE SUM <CR>
```

Point to and print the line containing the word SUM.

```
3        FORMAT (' THE SUM IS ',I5)
*CHANGE/SUM/RESULT/ <CR>
```

Change SUM to RESULT.

```
3        FORMAT (' THE RESULT IS ',I5)
*LOCATE (2I5) <CR>
```

Point to the line containing (2I5).

```
[*EOB*]
```

EDI reached end of buffer (EOB) without finding the text (2I5). The line pointer only moves forward through the buffer in response to a LOCATE command.

```
*TOP <CR>
```

Move the line pointer to the top of the buffer (one line before the first line of text).

```
* <CR>
        TYPE 1
```

Pressing carriage return in response to the * prompt instructs EDI to print the next line, which becomes the current line for editing. In this case, the next line is the first line in the buffer.

```
*EXIT <CR>
[EXIT]
>
```

Write the current buffer and the remainder of the input file to the output file, close both files, and exit to MCR.

Inserting and Deleting Text

This section illustrates how to use the following additional editor commands. Again, the optional portion of each command is underlined.

Command	Function
INSERT	Insert one or more new lines of text.
ADD	Append text to an existing line.
DELETE	Delete the current line.
<ESC>	Point to and print the previous line.

Command	Function
<u>RE</u>TYPE	Replace the current line with a new string of text.
<u>L</u>IST	Display at the terminal all lines remaining in the buffer, beginning with the current line. Note that the line pointer does not move.

Example:

The text following each break in this example explains the command that has just been given to the editor.

```
>EDI ADD.FTN <CR>
```

EDI retrieves the latest edited version of ADD.FTN.

```
[00009] LINES READ IN
[PAGE 1]
*INSERT <CR>
```

Switch to input mode. Insert the following line immediately before the first line of text in the buffer.

```
C       THIS PROGRAM ADDS TWO NUMBERS TOGETHER <CR>
<CR>
*LOCATE NUMBERS <CR>
J       FORMAT (' TYPE TWO NUMBERS - M,N')
*ADD !INPUT PROMPT <CR>
```

Append comment text to current line.

```
*PRINT <CR>
1       FORMAT (' TYPE TWO NUMBERS - M,N')!INPUT PROMPT
*LOCATE RESULT <CR>
3       FORMAT (' THE RESULT IS ',I5)
*DELETE <CR>
```

Delete the current line and move the pointer forward one line.

```
*<ESC>
```

Typing <ESC> in response to the * prompt instructs EDI to point to and print the previous line.

```
        TYPE 3,K+L
*INSERT <CR>
```

Enter input mode.

```
3       FORMAT (' THE SUM IS ',I5)!DISPLAY RESULT <CR>
```

Insert this new line of text.

```
    <CR>
```

Re-enter edit mode.

```
    *TOP <CR>
```

Reposition the line pointer at the top of the buffer.

```
    *<CR>
```

Point to and print the next line, which is the first line in the buffer.

```
C          THIS PROGRAM ADDS TWO NUMBERS TOGETHER
*RETYPE C          ADD DISPLAYS THE SUM OF TWO NUMBERS <CR>
```

Replace the current line with the text entered after the RETYPE command.

```
    *LIST <CR>
```

Display the remainder of the current block.

```
C          ADD DISPLAYS THE SUM OF TWO NUMBERS
           TYPE 1
1          FORMAT (' TYPE TWO NUMBERS - M,N')!INPUT PROMPT
           ACCEPT 2,K,L
2          FORMAT (2I5)
           TYPE 3,K+L
3          FORMAT (' THE SUM IS ',I5)!DISPLAY RESULT
           STOP
           END

*EXIT <CR>
[EXIT]
>
```

2.1.4 Basic EDI Commands

Table 2-1 summarizes a basic set of EDI commands. This set provides you with all the functions you need for simple editing. See the *RSX-11 Utilities Procedures Manual* for a full specification of EDI.

2.2 COMPILING A SIMPLE FORTRAN SOURCE PROGRAM

Compilation is the process of translating an ASCII source program into a machine-readable *object program*. In the call to the FORTRAN compiler, you supply a *file expression* that specifies the source program as an input file and the object program as an output file. For example:

```
>FOR ADD.OBJ=ADD.FTN
```

Table 2-1 Basic EDI Commands

Command	Command Format	Description
ADD	Add string	Append string to current line.
BOTTOM	BOttom	Move the line pointer to the bottom of the current block.
CHANGE	Change /string1/ string2/	Replace string1 with string2 in the current line.
<CR>	Carriage return	Print the next line, make it the new current line; or exit from input mode.
CTRL/Z	↑Z	Close the input and output files and terminate the editing session.
DELETE	Delete	Delete the current line.
<ESC>	ESCape or ALTmode	Point to and print the previous line.
EXIT	EXit	Close the input and output files and terminate the editing session.
INSERT	Insert [string]	Insert string on the next line or enter input mode if string is omitted.
LOCATE	Locate string	Locate the first line containing string. The search stops at the end of the current block.
NEXT	Next	Advance the line pointer to the next line.
PRINT	Print	Print the current line.
RENEW	RENew	Write the current block to the output file and read in a new block from the input file.
RETYPE	Retype string	Replace the current line with string.
TOP	Top	Move the line pointer to the top of the block.

This command string requests the FORTRAN compiler to compile the source program called ADD.FTN to produce an object file called ADD.OBJ. As the example file expression shows, you specify, from left to right, the output file (ADD.OBJ), an equal sign (=), then the input file (ADD.FTN).

Because the FORTRAN compiler defaults the object type to OBJ and the source type to FTN, you can abbreviate the above command string to

```
>FOR ADD=ADD
```

By appending *switches* to the input and/or the output file specifications, you can make special compilation requests. See the *IAS/RSX-11 FORTRAN IV User's Guide* or the *FORTRAN IV-Plus User's Guide* for a description of FORTRAN compiler switches. All the examples in this section assume you are using the compiler defaults.

Requesting a Listing File
In addition to translating a source program, the FORTRAN compiler can also supply a listing file on request. For example:

```
>FOR ADD,ADD=ADD <CR>
```

The second ADD in the output file list instructs the compiler to produce a listing file of the source program and the storage map. (The storage map lists all the symbolic names referred to in the program.) If the system supports the *line printer spooler*, the listing file is automatically directed to the line printer. (See Section 3.2.5 for a definition of spooling.) However, if your system does not spool line printer output, the command shown above creates a file called ADD.LST, stored on your system disk.

Note that you must insert a comma (,) between the object file name and the listing device name; in other words, the required format is

```
>FOR object,list=source
```

To send· the listing file to your own terminal, issue the following command string:

```
>FOR ADD,TI:=ADD[1]
```

You can also request the compiler to produce a source listing of a program without creating an object file as well. To make such a request, simply omit the object file field in the file expression. For example:

```
>FOR ,TI:=ADD <CR>
```

---

[1]If your system does not spool the listing file by default, you can request a line printer listing of the source file by substituting LP: for TI:, that is, >FOR ADD,LP:=ADD.

Note that you must enter a comma (,) before the listing device; otherwise the compiler interprets TI: as an object file name.

Requesting a Nonresident FORTRAN Compiler
If the message

```
MCR -- TASK NOT IN SYSTEM
```

appears after you issue the FOR command, the FORTRAN compiler is not currently resident in the system. In this case, issue the command

```
>RUN $FOR <CR>
```

This command requests the system to load the FORTRAN compiler into memory and to activate it. The compiler responds by prompting

```
FOR>
```

You then enter the file expression in the format illustrated above. For example:

```
>RUN $FOR <CR>
FOR>ADD,ADD=ADD <CR>
FOR><CTRL/Z>
```

After the compiler has completed your request, it displays its task prompt once more. Enter CTRL/Z to terminate the compiler and remove it from memory. Alternatively, you can enter a file expression to request another compilation and/or source listing.

2.3 ASSEMBLING A SIMPLE MACRO SOURCE PROGRAM
To assemble a MACRO source program, issue a command in the following format:

```
>MAC object,list=source
```

By appending switches to the input and/or the output file specifications, you can make special assembly requests. See the *IAS/RSX-11 MACRO-11 Reference Manual* for a description of assembly switches. All the examples in this section assume you are using the assembly defaults.

The Assembler defaults the object type to OBJ, the list type to LST, and the source type to MAC; therefore, the command

```
>MAC ADD,ADD=ADD
```

represents

```
>MAC ADD.OBJ,ADD.LST=ADD.MAC
```

The Assembler-generated listing includes an optional table of contents, an assembly listing, and a symbol table. To request a listing only, without an assembled object file, omit the object file name. For example:

```
>MAC ,ADD=ADD
```

See *Requesting a Listing File* in Section 2.2 for further information about obtaining a listing file.

Requesting a Nonresident Assembler
If the MACRO Assembler is not resident in memory when you issue a MAC command, the system displays the message

```
MCR -- TASK NOT IN SYSTEM
```

To load and activate the Assembler, issue the following command:

```
>RUN $MAC
MAC>
```

Then enter a file expression in response to the following Assembler prompt:

```
>RUN $MAC <CR>
MAC>ADD,ADD=ADD <CR>
MAC><CTRL/Z>
```

When the Assembler prompts for further input, either enter a file expression to request another compilation and/or source listing, or type CTRL/Z to terminate the Assembler.

2.4 LINKING OBJECT PROGRAMS – THE TASK BUILDER
The Task Builder is a system program that makes an executable program unit, called a task, from one or more object modules; it links the object modules, resolves any references to the system library, and produces a single task image. The task image resides on disk until you issue an MCR command to install and run it.

The simplest form of Task Builder (TKB) command input is

>TKB task-image=object

where task-image, the output file, is the name given to the executable task; and object, the input file, is the name of a module produced by a compiler or assembler. For example:

```
>TKB ADD.TSK=ADD.OBJ <CR>
```

Because TKB defaults the output file type to TSK, and the input file type to OBJ, you can shorten the above command string to

```
>TKB  ADD=ADD  <CR>
```

2.4.1 Full TKB Command Line

The format of a full TKB command line accommodates three output files and any number of input files. The three output files are as follows:

1.   The *task image* (name.TSK), which contains the executable task image
2.   A *memory allocation file* (name.MAP), which lists information about the size and location of components within the task
3.   A *symbol definition file* (name.STB), which contains information about the task's global symbol definitions

The list of input files comprises the one or more object modules to be linked into a single task image.

The format is

>TKB task-image,map,symbol-def=input,...

If you omit an output file at the beginning or middle of the output list, retain a comma in the file's place to specify a null field. On the right-hand side of the equal sign, use a comma to separate one input file from another. Table 2-2 illustrates all the possible requests for TKB output files.

Table 2-2   TKB Output File Requests

Command Line	Requested Output Files
>TKB ADD,MAP,SYMBOL=ADD	All three output files
>TKB ,,SYMBOL=ADD	Symbol definition only
>TKB ,MAP,SYMBOL=ADD	Memory allocation and symbol definition
>TKB ,MAP=ADD	Memory allocation only
>TKB ADD,,SYMBOL=ADD	Task image and symbol definition
>TKB ADD,MAP=ADD	Task image and memory allocation
>TKB ADD=ADD	Task image only

2.4.2 Simple Multiline Format

A long list of input files can cause a TKB command line to exceed your terminal's maximum line length. If this is the case, invoke TKB in the following manner:

```
>TKB <CR>
TKB>
```

When you terminate the line immediately after entering TKB, MCR activates the Task Builder, which then prompts for input. TKB continues to display its prompt after each line of input until it receives the terminating characters // in the first two character positions. For example, the single command line

```
>TKB TASK,MAP,SYMBOL=INPUT1,INPUT2,INPUT3 <CR>
```

can be entered as follows:

```
>TKB <CR>
TKB>TASK,MAP,SYMBOL=INPUT1 <CR>
TKB>INPUT2,INPUT3 <CR>
TKB>// <CR>
>
```

2.4.3 Default Types

The default types in the TKB command line are as follows:

File	Type
Task image	TSK
Memory allocation	MAP
Symbol definition	STB
Input	OBJ

Therefore, by default, the command line

```
>TKB ADD,ADD,ADD=ADD
```

represents

```
>TKB ADD.TSK,ADD.MAP,ADD.STB=ADD.OBJ
```

2.4.4 Listing the Memory Allocation (MAP) File

The memory allocation (also called map) file is an ASCII file that contains information about the size and location of components within the task. If your system supports the spooling option, TKB spools this file to the line printer by default.

However, without the spooling option, TKB stores this file on your system disk. If you want a listing only of the map file, rather than a stored

disk image, insert the name of a listing device in the place of the map file name. For example:

```
>TKB ADD,LP:,SY:ADD=ADD
```

This command instructs TKB to list the file called ADD.MAP on the line printer and to store the files ADD.TSK and ADD.STB on your system disk.

Note that the third output file is called SY:ADD rather than ADD. SY: is the device name for your system disk. You must prefix the name of the STB file with SY: because the preceding device name has changed the default device to LP:. The command

```
>TKB ADD,LP:,ADD=ADD
```

instructs TKB to list both the map and the symbol definition file on the line printer, even though the third output file does not specify LP:. Because the STB file is in machine-readable code rather than ASCII code, an STB listing is of little or no use to you. The effect of the LP: specification does not cross the equal sign; =ADD still represents SY:ADD.OBJ.

2.4.5 Task Builder Switches and Options

The Task Builder includes a variety of switches and options that afford you additional control over the construction of a task image. These switches and options are described in the *RSX-11M Task Builder Reference Manual*.

2.5 RUNNING A TASK

The goal of the procedures described so far is to produce a program unit, called a task, that can run on RSX-11M. Linking, the last step in the preparation sequence, has generated a task image file (ADD.TSK) stored on your system disk. To run the task, issue the MCR Run command, specifying ADD.TSK as a parameter:

```
>RUN ADD <CR>
```

(The Run command defaults the type to TSK.) This command instructs the system to perform the following functions:

- Locate ADD.TSK on your system disk.
- Load a copy of the task image into memory.
- Execute the task.
- Remove the memory-resident task when it has completed its run.

The task image file remains on disk, ready to be run, until you explicitly delete it.

The following listing illustrates three consecutive runs of the FORTRAN program ADD.

```
>RUN ADD <CR>
TYPE TWO NUMBERS - M,N
7,3 <CR>
THE SUM IS    10
TT13 -- STOP
>
>RUN ADD <CR>
TYPE TWO NUMBERS - M,N
522,628 <CR>
THE SUM IS 1150
TT13 -- STOP
>
>RUN ADD <CR>
TYPE TWO NUMBERS - M,N
9,16 <CR>
THE SUM IS    25
TT13 -- STOP
>
```

Note that, at the completion of each run, the program displays the line

```
TT13 -- STOP
```

TT13 is a temporary name assigned to the task while it is running. The version of the Run command shown in these examples causes the system to name the task TTn, where n is the unit number of the requesting terminal.

Figure 2-2 reviews the steps in creating the FORTRAN task ADD, showing the specific files created along the way.

Source
Program ADD.FTN

>FOR   Listing

COMPILER → ADD.LST

Object
Module ADD.OBJ

                              USER
                              LIBRARIES

>TKB   SYSTEM
                              LIBRARIES

TASK
BUILDER   Map

                              ADD.MAP

Task
Image ADD.TSK   Symbol Definition

>RUN   ADD.STB

EXECUTING
TASK
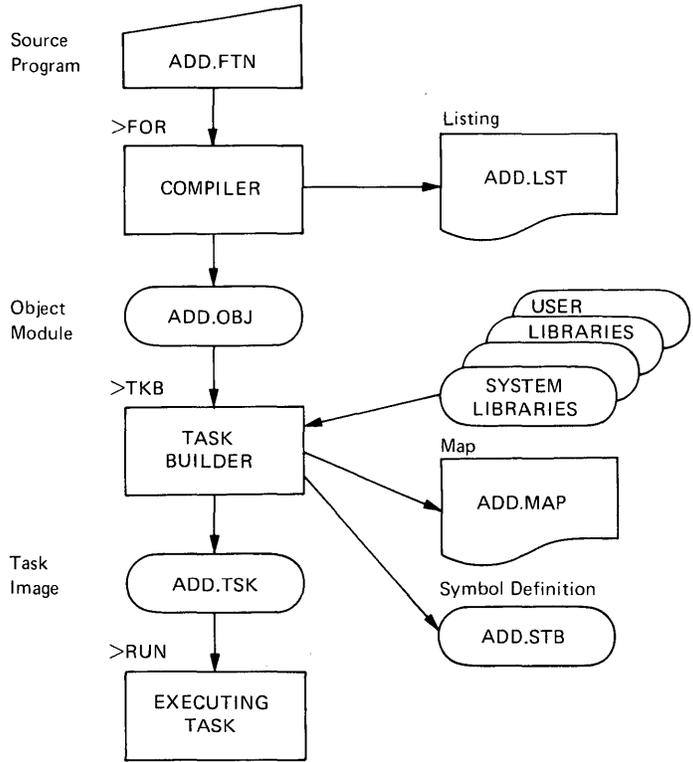
Figure 2-2   Steps in Creating the FORTRAN Task ADD

CHAPTER 3
THE FILES

This chapter describes files and file handling in two parts. Section 3.1 defines all the elements of a complete file specification. Section 3.2 then explains how to use a system program to manipulate files.

3.1 COMPLETE FILE SPECIFICATIONS

The examples given in Chapters 1 and 2 have stressed the use of defaults in a command line. Defaults are useful to know because they allow you to type as little as necessary when providing input to the system. However, there are many situations in which you must supply a complete file specification because defaults do not apply. Section 2.1.1, *Creating a Source Program File*, defines two fields of a file specification: the file name and the file type. The three additional fields required to complete a file specification are

> dev: The name of the device that holds the volume on which the file resides

> [g,m] A User Identification Code (UIC) that identifies the directory that lists the file

> ;n A number that differentiates among various versions of the same file

The format of a complete file specification is

> dev:[g,m]filename.type;n

Examples:

> MT1:[116,23]DATA.DAT;4
> DT:[203,204]TASK.TSK;1
> DK2:[34,63]CALC.MAP;2

3.1.1 The Device Name

The device name specifies the unit on which the file resides. The name consists of 2 alphabetic characters, an optional 1- or 2-digit octal unit number, followed by a colon (:). When the name does not include a unit number, the system assumes that the unit number is 0. Example device names and corresponding units are listed on the following page.

Name	Corresponding Unit
DK2:	RK05 disk, unit 2
DT1:	DECtape, unit 1
MT:	TU10 magnetic tape, unit 0

The device name can specify either an actual physical unit, like the three listed above, or a *pseudo device* name, which corresponds to different physical units, according to whoever enters the name. For example, the name TI: refers to the terminal at which you are entering input. Another pseudo device name is SY:, which corresponds to your default system disk. All the files you have created so far reside on SY:.

Table 3-1 lists the names of commonly used devices. The letter n stands for unit number. There are different types of disks and magnetic tapes, which have correspondingly different device names. Ask someone at your installation which names you should use. See the *RSX-11M Operator's Procedures Manual* for a complete list of device names.

Table 3-1   Commonly Used Device Names

Device	Name
DECtape	DTn:
Disk	DBn: DKn: DMn: DPn: DXn:
Line Printer	LPn:
Magtape	MMn: MTn:
Pseudo input terminal	TI:
Terminal	TTn:
System default device	SY:

Because the names TI:, TT:, and LP: represent input and/or output devices, rather than storage media, they generally do not appear in a complete file specification. When you refer to one of these devices in a file expression, the device name stands alone. For example, the following command sends a copy of ADD.FTN on your system disk to your terminal:

```
>PIP TI:=ADD.FTN
```

3.1.2 User File Directories (UFDs)

When you logged on (see Section 1.2.2), you either directly or indirectly (through your last name) specified your UIC. This UIC identifies the default *User File Directory (UFD)* set up for you on SY: when the system manager made you an authorized user. This UFD is itself a file that lists the names of all files in your area. The system cannot locate a file unless it knows the UFD in which it is listed.

If your UIC is [203,125], the name of your default UFD is 203125.DIR;1. However, when referring to a UFD within a file specification, you must use the UIC format, namely [203,125].

Your default UFD is not the only directory you can access. On many occasions you may need to access files belonging to someone else. In this event, you can use an MCR command to change your default UFD (SET/UIC=[g,m]; see the *RSX-11M Operator's Procedures Manual*), or you can explicitly specify a different UFD in a file specification. (Note that neither of these actions changes the UIC with which you logged on.)

Suppose you need to obtain a copy at your terminal of a file (LABEL.MAC, for example) listed in UFD [203,200]:

```
>PIP TI:=[203,200]LABEL.MAC
```

This command assumes that LABEL.MAC resides on your SY:.

In a system that does not support multiuser protection, your default UFD corresponds to the UIC specified in the last SET /UIC=[g,m] command issued from your terminal. To obtain a display of the current default UFD, issue the command

```
>SET /UIC <CR>
```

The system responds as follows:

```
UIC=[g,m]
```

where [g,m] is the current default UFD.

Restrictions on File Access

The system does impose restrictions on access to UFDs and other files that belong to other users. Every file includes a *protection mask* that determines who can access the file and in what manner; when you attempt to access a file, the system checks your UIC against the file's protection mask. If the check fails, your attempted access fails; and the system returns an error message at your terminal.

For example, if you attempt to edit a file that prohibits access by your UIC, EDI displays the following message:

```
EDI -- PRIVILEGE VIOLATION
```

See the *RSX-11M Operator's Procedures Manual* for a more detailed description of file protection.

3.1.3 Version Numbers

A version number is octal and in the range 0 to 77777. When a file is created, the system assigns it a version number of 1. Subsequently, the system increments the version number by 1 each time a new version of the same file is created. The "same file" is one that has the same name and type and is listed in the same UFD on the same volume.

For example, whenever you edit a file, EDI creates an output file to contain the edited file. The file specification that refers to the output file is identical to the input file specification, except that the version has been increased by 1.

Input File   **Output File**

FILE.FTN;1 ⟶ EDI ⟶ FILE.FTN;2
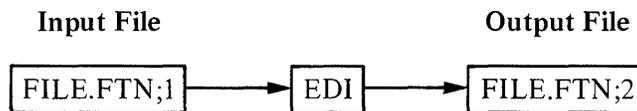
Recompilations of the same source file also produce files with the same specifications except for differing version numbers. For example, the command

    >FOR ADD=ADD

results in the creation of a file called ADD.OBJ;1. If you repeat the command, the resultant file is called ADD.OBJ;2 since the file ADD.OBJ;1 already exists.

Input File   **Output File**

① ADD.FTN;1 ⟶ FOR ⟶ ADD.OBJ;1

② ADD.FTN;1 ⟶ FOR ⟶ ADD.OBJ;2
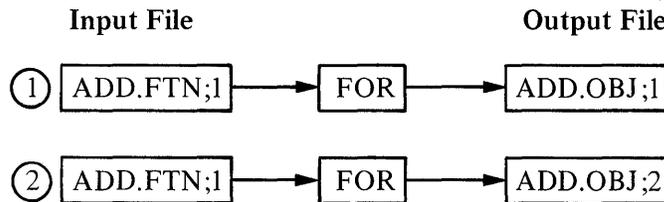
3.1.4 File Specification Defaults

Table 3-2 explains the defaults for each field of the file specification.

Table 3-2 File Specification Defaults

Field	Default
dev:	SY:
[g,m]	The UIC with which you logged on, or a UIC determined by the MCR command SET /UIC=[g,m] (see the *RSX-11M Operator's Procedures Manual*).
filename	No default.
type	Depends on the command string in which the file specification appears.
version	For input files, the highest existing version. For output files, the highest existing version plus 1. Note that some commands require an explicit version number.

3.2 MANIPULATING FILES USING PIP

In the process of creating, editing, compiling, and linking your program, you have generated a number of disk files:

- The original source file (ADD.FTN;1)
- An edited version of the source file (ADD.FTN;2)
- An object file containing the compiled program (ADD.OBJ;1)
- A task image file containing linked object code (ADD.TSK;1)
- A memory-allocation file (ADD.MAP;1)
- A symbol-definition file (ADD.STB;1)
- Additional versions of the above files, if you have repeated any of the program preparation steps

All of these files remain on SY:, taking up space, until you explicitly delete them. Once you have determined which file you need to keep, how do you eliminate those you do not want?

The following sections describe how to use the Peripheral Interchange Program (PIP). This program allows you not only to delete obsolete files, but also to perform a variety of other file-manipulation functions. When you invoke PIP, it performs a specific function according to the parameters you supply. (See the *RSX-11 Utilities Procedures Manual* for a complete description of PIP; this manual describes only a subset of PIP functions.)

3.2.1 Listing a UFD

All the files you have created are listed in your UFD. The following PIP command displays that list at your terminal.

```
>PIP /LI <CR>
```

/LI is the List *switch*, where a switch is a slash followed by 2 letters. The letters abbreviate, or otherwise identify, the switch's function. For example, /LI means "list"; /DE means "delete."

If you plan to issue a series of PIP commands, you can invoke PIP as follows:

```
>PIP <CR>
PIP>/LI <CR>
```

```
DIRECTORY DB0:[301,314]
25-MAY-77 14:48

ADD.OBJ;2          2.              25-MAY-77 14:31
ADD.TSK;1         29.         C   20-MAY-77 14:32
ADD.MAP;1          4.              25-MAY-77 14:32
ADD.STB;1          3.              25-MAY-77 14:32
ADD.FTN;1          1.              20-MAY-77 15:26
ADD.TSK;3         29.         C   25-MAY-77 14:33
ADD.OBJ;1          2.              20-MAY-77 15:27
ADD.TSK;2         29.         C   20-MAY-77 15:28

TOTAL OF 99./112. BLOCKS IN 8. FILES

PIP>
```

After listing the directory, PIP displays its task prompt and waits for further input. To terminate PIP, type CTRL/Z in response to the prompt

```
PIP><CTRL/Z>
>
```

The PIP directory listing includes the following information:

A The physical device unit on which the files are stored, and the UIC that owns the directory. The unit named is your SY:; the UIC is your current UIC.

B The date and time you issued the PIP request.

C The name, type, and version number of each file.

D The size of each file in blocks. A block is 512 bytes (256 words) long.

(E) The letter C if the file is contiguous, that is, not split into physically separated sectors on the disk.

(F) The date and time each file was created.

(G) The cumulative size in blocks of all the files listed and the number of blocks allocated for all the files.

Figure 3-1 assigns the letters in the above list to the appropriate parts of a directory listing.

```
                        (A)
                 DIRECTORY DB0:[301,314]
          (B)    25-MAY-77 14:48
                      (C)              (D)      (E)       (F)
                 ADD.OBJ;2         2.                25-MAY-77 14:31
                 ADD.TSK;1         29.        C       20-MAY-77 14:32
                 ADD.MAP;1         4.                25-MAY-77 14:32
                 ADD.STB;1         3.                25-MAY-77 14:32
                 ADD.FTN;1         1.                20-MAY-77 15:26
                 ADD.TSK;3         29.        C       25-MAY-77 14:33
                 ADD.OBJ;1         2.                20-MAY-77 15:27
                 ADD.TSK;2         29.        C       20-MAY-77 15:28

          (G)    TOTAL OF 99./112. BLOCKS IN 8. FILES
```

Figure 3-1   Sample Directory Listing

PIP also accepts switches that request either fewer or more details about the files listed; these switches are described in the *RSX-11 Utilities Procedures Manual*.

3.2.2   Listing Specific File Information

PIP also allows you to obtain information about one file or a specific group of files. For example, if you want to see how many versions of ADD.TSK exist in your directory, issue the command

```
>PIP ADD.TSK;*/LI  <CR>
```

As the example illustrates, the file specification always precedes the switch.

The command requests PIP to display a directory listing of all versions of the file called ADD.TSK. The PIP request refers to all versions of the file because you entered an asterisk (*) in the version number field of the file specification. The asterisk, called a *wildcard*, stands for "all"; it can be placed in any field of the file specification, other than the device field. For example,

```
ADD.*;*
```

means all types and all versions of the file name ADD. Since all the files in the directory listed above are named ADD, the command

```
>PIP ADD.*;*/LI <CR>
```

has the same effect as

```
>PIP /LI <CR>
```

which lists out all files in the directory.

3.2.3 Deleting Files
PIP includes two switches for deleting files:

/DE   Delete the one or more files specified.

/PU   Purge (delete) all but the highest version(s) of the one or more files specified.

Once you know what files are listed in your UFD, you can decide which files you want to delete. Considering the list shown in Figure 3-1, suppose you decide to retain only the highest version of each file. You consequently need to delete the following files:

ADD.OBJ;1
ADD.TSK;1
ADD.TSK;2

To delete these files by means of the /DE switch, issue the command

```
>PIP ADD.OBJ;1,ADD.TSK;1,ADD.TSK;2/DE <CR>
```

This example demonstrates how to enter a list of specifications; a comma separates one specification from the next, and the switch follows the last specification (PIP filespec,..., filespec/sw).

Note that the /DE option requires either an explicit version number or a wildcard in the version field. However, the wildcard (*) feature is inappropriate when the UFD lists other files with the same name, but with different types and versions that you want to retain. In this instance, it is necessary to specify each file to be deleted.

When you want to eliminate all but the highest version of one or more files, the /PU switch is more efficient than /DE. The following command has the same effect as the above use of the /DE switch (in the context of the UFD listed in Figure 3-1):

```
>PIP ADD.*/PU <CR>
```

/PU does not affect files that have only one version listed within the UFD. Note that the purged file specification does not include a version field.

```
>PIP /LI <CR>

DIRECTORY DB0:[301,314]
25-MAY-77 14:56

ADD.OBJ;2            2.           25-MAY-77 14:31
ADD.MAP;1           4.           25-MAY-77 14:32
ADD.STB;1           3.           25-MAY-77 14:32
ADD.FTN;1           1.           20-MAY-77 15:26
ADD.TSK;3          29.      C    25-MA7-77 14:33

TOTAL OF 39./49. BLOCKS IN 5. FILES
```

As the listing shows, the directory now contains only the highest version of each file.

3.2.4 Copying Files

Copying files is PIP's default function; that is, PIP assumes the copy function when you do not specify a switch. For example, the following command copies file ADD.MAP from your area on SY: to your area on another disk:

```
>PIP DK:=ADD.MAP <CR>
```

The command includes the call to PIP, followed by a file expression (see Section 2.2) in the form

> outfile=infile

where infile is the file to be copied and outfile is the destination of the new copy. When outfile omits the UFD, file name, type, and/or version, PIP defaults the UFD, filename, type, and version to the equivalent fields in the infile.

Any volume that you reference in a file specification must be allocated to you (multiuser protection systems only) and mounted. If the volume specified or implied in either infile or outfile is not mounted, PIP returns the message

```
PIP -- DEVICE NOT MOUNTED
```

In addition, before you can copy a file to a directory on another volume, the directory must exist. In a multiuser protection system, your directory on SY: is the only UFD automatically created for you; and in a non-multiuser system, no UFDs are automatically created. If the output disk

specified in the above example does not contain a UFD corresponding to your UIC, PIP returns the message

```
PIP -- CANNOT FIND DIRECTORY FILE
```

Device allocation, mounting, and the creation of UFDs are all MCR functions, effected respectively by the following commands:

- ALLOCATE
- MOUNT
- UFD

See the *RSX-11M Operator's Procedures Manual* for a description of these commands. Specifically note the sections on nonprivileged users working in a multiuser system. Note, however, that these three commands cannot be issued by nonprivileged users if the system does not support multiuser protection.

3.2.5 Spooling Files to the Line Printer

System users often need line printer listings of files; consequently, the line printer is usually in heavy demand. RSX-11M provides an optional feature called *spooling* that allows more efficient, nondisruptive use of the line printer. To spool files to the line printer, use the PIP switch /SP. For example:

```
>PIP ADD.MAP/SP    <CR>
```

Note that you do not need to specify LP: as the output file. This command requests PIP to enter the name ADD.MAP in a queue of files waiting to be listed at the printer. The system maintains the queue of names, copying each file or group of files to the line printer on a first-queued, first-printed basis.

Find out from the system manager whether or not your system supports spooling. If it supports spooling, the system automatically directs certain files to the line printer (see *Requesting a Listing File*, Section 2.2). Also note that spooling can be generated to delete all files spooled to the printer; this feature should therefore be used with care.

If you issue a command like the one above, and your system does not support the spooling option, PIP returns the message

```
PIP -- FAILED TO SPOOL FILE FOR PRINTING
```

If this message appears, you must request a listing in the conventional copy format:

```
>PIP LP:=ADD.MAP <CR>
```

3.2.6 Renaming Files

The PIP switch /RE allows you to rename existing files. For example, the command

```
>PIP ADDTWO.*;*=ADD.*;*/RE  <CR>
```

tells PIP to change the file-name field of all types and versions of the files named ADD to ADDTWO. Note that you must explicitly specify either a number or a wildcard in both version fields when you use the /RE switch. The wildcards in the output file specifier indicate that the type and version of the renamed files are not to be changed. Your directory now looks as follows:

```
DIRECTORY DB0:[301,314]
25-MAY-77 16:43

ADDTWO.OBJ;2        2.          25-MAY-77 14:31
ADDTWO.MAP;1        4.          25-MAY-77 14:32
ADDTWO.STB;1        3.          25-MAY-77 14:32
ADDTWO.FTN;1        1.          20-MAY-77 15:26
ADDTWO.TSK;3       29.      C   25-MAY-77 14:33

TOTAL OF 39./49. BLOCKS IN 5. FILES
```

Because the renaming function does not transfer data, you cannot specify a different device in the output file specification. If you want to rename a file as you copy it to another volume, enter the new name in the output specification of the copy command line. For example:

```
>PIP DK:ADDTWO.*;*=ADD.*;*  <CR>
```

This command tells PIP to copy all types and versions of the file named ADD, which are stored in your UFD on SY:, to an equivalent UFD on the DK disk, unit 0, where they are named ADDTWO.

APPENDIX
SUMMARY OF EXAMPLE PROGRAM

The following listing summarizes the development of the FORTRAN
program ADD and the manipulation of the resultant files.

```
>HEL <CR>
ACCOUNT OR NAME: CHARLES <CR>
PASSWORD: GREY <CR>

          RSX-11M BL18 MULTI-USER SYSTEM

GOOD MORNING
22-MAY-77 11:07 LOGGED ON TERMINAL TT4:

22-MAY-77

          SYSTEM WILL BE DOWN TODAY FROM 13:00-15:00 FOR
          CORRECTIVE MAINTENANCE

>EDI ADD.FTN <CR>
[CREATING NEW FILE]
INPUT
          TYPE 1 <CR>
1         FORMAT (' ENTER TWO NUMBERS - M,N') <CR>
          APPE\EPP\CCEPT 2,K,L <CTRL/R>
          ACCEPT 2,K,L <CR>
2         FORMAT (22\2\I5) <CR>
          PRINT ^U
          TYPE 3,K+L <CR>
3         FORMAT (' THE SUM IS ',I5) <CR>
          STOP <CR>
          END <CR>
<CR>
*EX <CR>
[EXIT]
>PIP TI:=ADD.FTN <CR>

          TYPE 1
1         FORMAT (' ENTER TWO NUMBERS - M,N')
          ACCEPT 2,K+L
2         FORMAT (2I5)
          TYPE 3,K+L
3         FORMAT (' THE SUM IS ',I5)
          STOP
          END
>EDI ADD.FTN <CR>
[00009 LINES READ IN]
[PAGE 1]
*LOCATE ENTER <CR>
1         FORMAT (' ENTER TWO NUMBERS - M,N')
*CHANGE/ENTER/TYPE/ <CR>
1         FORMAT (' TYPE TWO NUMBERS - M,N')
*NEXT <CR>
*PRINT <CR>
          ACCEPT 2,K+L
```

```
*LOCATE SUM <CR>
3         FORMAT (' THE SUM IS ',I5)
*CHANGE/SUM/RESULT/ <CR>
3         FORMAT (' THE RESULT IS ',I5)
*LOCATE (2I5) <CR>
[*EOB*]
*TOP <CR>
*<CR>
          TYPE 1
*EXIT <CR>
[EXIT]
>EDI ADD.FTN <CR>
[00009] LINES READ IN
[PAGE 1]
*INSERT <CR>
C         THIS PROGRAM ADDS TWO NUMBERS TOGETHER <CR>
<CR>
*LOCATE NUMBERS <CR>
1         FORMAT (' TYPE TWO NUMBERS - M,N')
*ADD !INPUT PROMPT <CR>
*PRINT <CR>
1         FORMAT (' TYPE TWO NUMBERS - M,N')!INPUT PROMPT
*LOCATE RESULT <CR>
3         FORMAT (' THE RESULT IS ',I5)
*DELETE <CR>
*<ESC>
          TYPE 3,K+L
*INSERT <CR>
3         FORMAT (' THE SUM IS ',I5)!DISPLAY RESULT <CR>
<CR>
*TOP <CR>
*<CR>
C         THIS PROGRAM ADDS TWO NUMBERS TOGETHER
*RETYPE C         ADD DISPLAYS THE SUM OF TWO NUMBERS <CR>
*LIST <CR>
C         ADD DISPLAYS THE SUM OF TWO NUMBERS
          TYPE 1
1         FORMAT (' TYPE TWO NUMBERS - M,N')!INPUT PROMPT
          ACCEPT 2,K,L
2         FORMAT (2I5)
          TYPE 3,K+L
3         FORMAT (' THE SUM IS ',I5)!DISPLAY RESULT
          STOP
          END

*EXIT <CR>
[EXIT]
>FOR ADD,ADD=ADD <CR>
>TKB ADD,ADD,ADD=ADD
>RUN ADD <CR>
TYPE TWO NUMBERS - M,N
7,3 <CR>
THE SUM IS    10
TT13 -- STOP
>
>RUN ADD <CR>
TYPE TWO NUMBERS - M,N
522,628 <CR>
THE SUM IS 1150
TT13 -- STOP
>
>RUN ADD <CR>
TYPE TWO NUMBERS - M,N
9,16 <CR>
```

```
THE SUM IS   25
TT13 -- STOP
>PIP <CR>
PIP>/LI <CR>



DIRECTORY DB0:[301,314]
25-MAY-77 14:48

ADD.OBJ;2            2.              25-MAY-77 14:31
ADD.TSK;1           29.        C    20-MAY-77 14:32
ADD.MAP;1            4.              25-MAY-77 14:32
ADD.STB;1            3.              25-MAY-77 14:32
ADD.FTN;1            1.              20-MAY-77 15:26
ADD.TSK;3           29.        C    25-MAY-77 14:33
ADD.OBJ;1            2.              20-MAY-77 15:27
ADD.TSK;2           29.        C    20-MAY-77 15:28

TOTAL OF 99./112. BLOCKS IN 8. FILES

PIP>
PIP><CTRL/Z>
>PIP ADD.*/PU <CR>
>PIP /LI <CR>

DIRECTORY DB0:[301,314]
25-MAY-77 14:56

ADD.OBJ;2            2.              25-MAY-77 14:31
ADD.MAP;1            4.              25-MAY-77 14:32
ADD.STB;1            3.              25-MAY-77 14:32
ADD.FTN;1            1.              20-MAY-77 15:26
ADD.TSK;3           29.        C    25-MA7-77 14:33

TOTAL OF 39./49. BLOCKS IN 5. FILES

>PIP ADDTWO.*;*=ADD.*;*/RE   <CR>
>PIP /LI <CR>

DIRECTORY DB0:[301,314]
25-MAY-77 16:43

ADDTWO.OBJ;2         2.              25-MAY-77 14:31
ADDTWO.MAP;1         4.              25-MAY-77 14:32
ADDTWO.STB;1         3.              25-MAY-77 14:32
ADDTWO.FTN;1         1.              20-MAY-77 15:26
ADDTWO.TSK;3        29.        C     25-MAY-77 14:33

TOTAL OF 39./49. BLOCKS IN 5. FILES
```

INDEX

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.

digital

digital equipment corporation