RSX–11M
Beginner's Guide

Order No. AA-5245B-TC

RSX-11M V3.2

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, June 1977
Revised, June 1979

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

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.

The following are trademarks of Digital Equipment Corporation:

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-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

CONTENTS

CONTENTS (Cont.)

FIGURES

TABLES

CONTENTS (Cont.)

Page

TABLES (Cont.)

PREFACE

MANUAL OBJECTIVES

The information in this manual enables the new user of the RSX-11M operating system to create a text file, using an editing program, and to create and run a simple FORTRAN-IV program from that file.

INTENDED AUDIENCE

This manual is for the new RSX-11M user. It assumes that you have some computer experience, but no experience with RSX-11M.

STRUCTURE OF THIS DOCUMENT

This manual is structured so you can learn RSX-11M concepts and terminology as you create and run a FORTRAN-IV program called ADD.FTN.

Chapter 1 explains how to use a terminal on the RSX-11M operating system. Chapter 2 discusses program preparation on RSX-11M, including use of an editing program, a compiler, and the Task Builder. Chapter 3 outlines the RSX-11M method of handling files and records. Chapter 4 discusses two system programs designed for handling files: PIP and the Queue Manager. Appendix A outlines the steps involved in the development of ADD.FTN, without the extensive explanation provided in earlier chapters. (For more detailed information on programming under RSX-11M, see the *RSX-11M/M-PLUS Guide to Program Development.*)

ASSOCIATED DOCUMENTS

This manual refers frequently to other RSX-11M documents which are described in the *RSX-11M Documentation Directory.* The directory defines the intended audience for the manuals in the RSX-11M set and provides a synopsis of each manual's contents.

CONVENTIONS USED IN THIS MANUAL

"print"	The term "print" indicates what the computer displays at the user's terminal.
"type"	The term "type" indicates what the user types at the terminal keyboard.
Black ink	In illustrations of dialogue between the user and the computer, what the computer prints is shown in black ink.
Red ink	In illustrations of dialogue between the user and the computer, what the user types is shown in red ink.

[ ]

Square brackets enclose the optional parameters to a command. For example:

```
>PRINT [/optional switches]
```

However, the brackets are not part of the command. That is, if you want to use the optional COPIES switch with the print command above, you do not include the brackets in the command line. You simply type:

```
>PRINT /COPIES:2 FOO.FUM
```

[144,34]

A UFD specification in a command line is always enclosed in brackets when you type it; that is, the brackets are required for the UFD syntax.

RUN

Uppercase characters, in lines illustrating command formats, indicate material that you must type exactly as shown.

filename

Lowercase characters, in lines illustrating command formats, indicate variable parameters that you select.

CHAPTER 1
USING THE TERMINAL

This chapter tells you how to use a terminal to communicate with the RSX-11M operating system. The operating system, in turn, communicates with the computer hardware.

An operating system translates language you can understand into language the computer can understand. It also translates messages from the computer hardware to you. This two-way (interactive) communication process usually takes place through a terminal.

You can use a variety of terminals with RSX-11M. These fall into two general categories: hardcopy terminals that print characters on continuous forms of paper, and Cathode Ray Tube (CRT) terminals that display characters on a screen. Most CRT terminals do not provide permanent records of the activity on your terminal. Figure 1-1 shows an LA36 hardcopy terminal. Figures 1-2 and 1-3 show the VT52 and the VT100 CRT terminals.

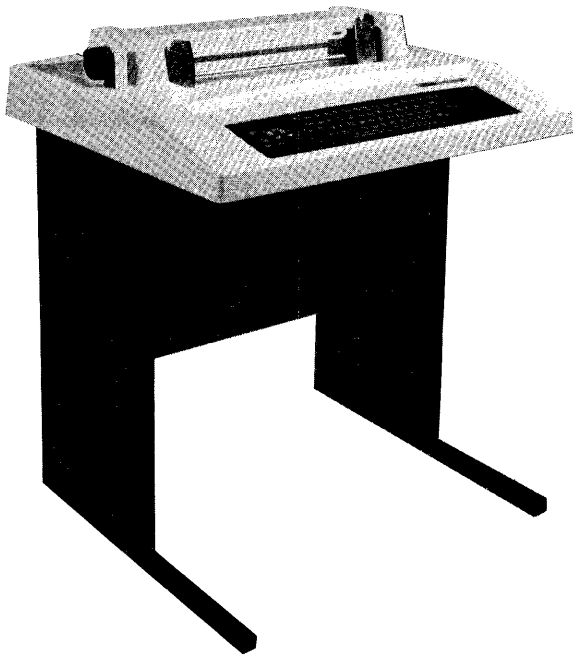

Figure 1-1 An LA36 Terminal

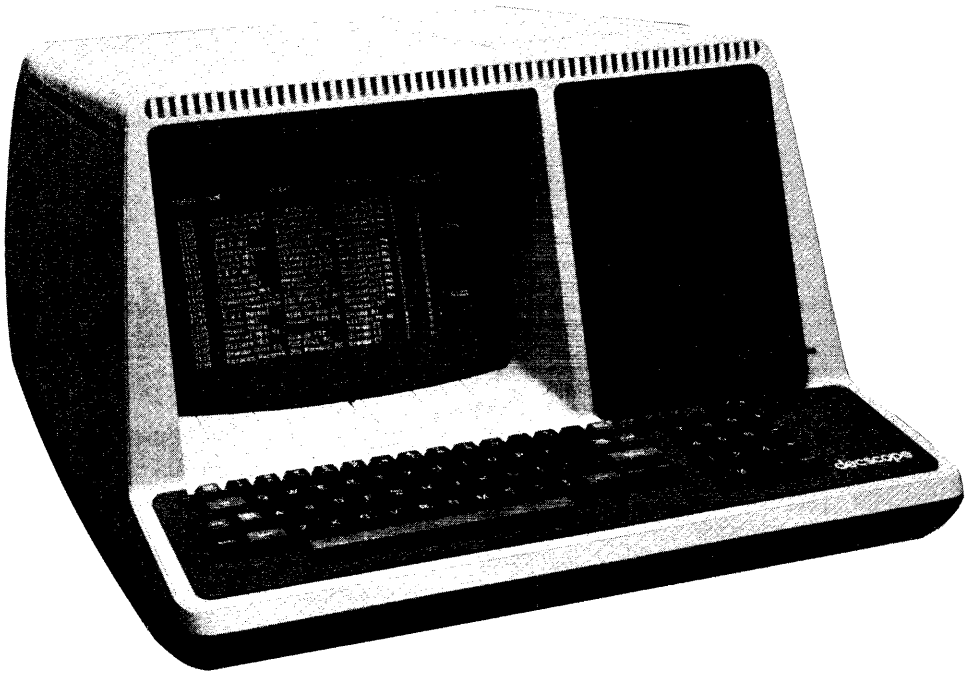

Figure 1-2 A VT52 Terminal

Figure 1-3 A VT100 Terminal

1.1 USING THE TERMINAL KEYBOARD

All terminal keyboards have the same basic layout as a typewriter. In addition, the keyboard includes special keys that affect various computer functions. These keys are not in the same position on all terminals; so check the keyboard layout each time you use a terminal that is new to you. Figure 1-4 is the keyboard of the LA36 and VT52 terminals. Figure 1-5 illustrates the keyboard of the VT100 terminal.

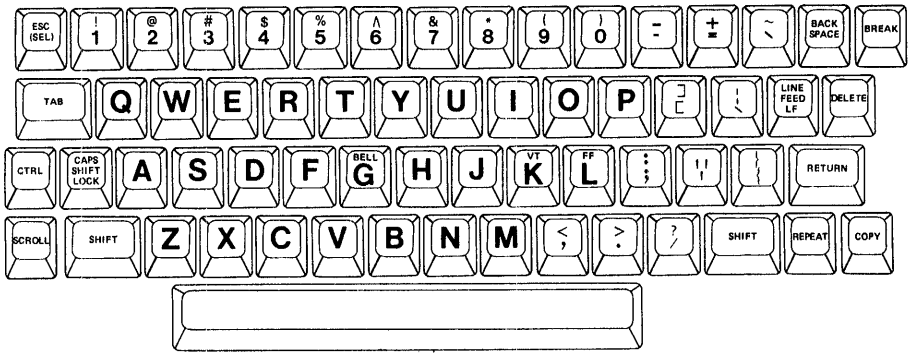

Figure 1-4 LA36/VT52 Terminal Keyboard

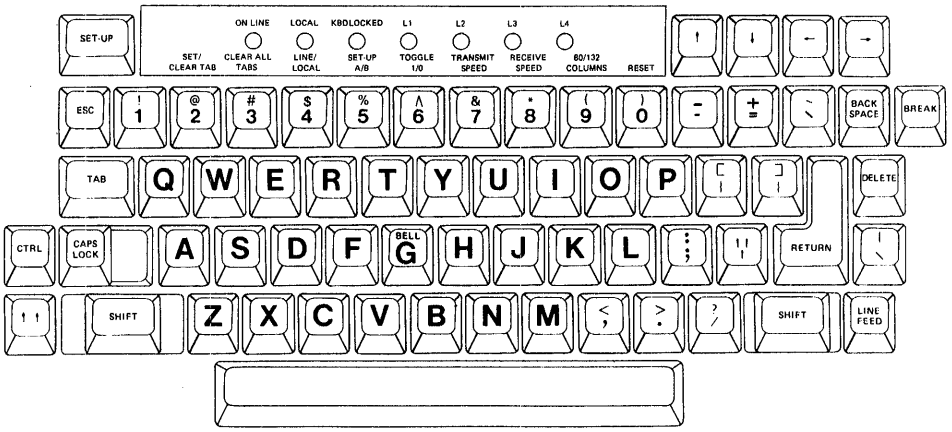

Figure 1-5 VT100 Terminal Keyboard

1.1.1 Terminal Function Keys

Some of the function keys that appear on terminal keyboards are described in Table 1-1. Check the documentation for the system program you are using for specific information on that program's use of terminal function keys.

1.1.2 Control Characters

You can enter a control character by pressing the appropriate letter while you hold down the CTRL key. The system responds to some control characters by displaying the up arrow (^) followed by the letter you typed (^U, for example). Some other control characters are not echoed on your terminal, depending on the function they perform.

The control keys listed below are options selected at system generation. Therefore, if they do not work for you as described below, check with your system manager to see if they are part of your system.

Table 1-1 Special Terminal Keys

Function Key	Description
CR or RETURN	Terminates an input line and advances the carriage or cursor to the first position on the next line.
CTRL/n	Part of numerous 2-key combinations (CTRL and a letter key. When you press the CTRL key, followed by the appropriate letter, the combinations can perform a variety of functions. Each valid combination is called a control character, represented in this manual by n, where n is a variable letter (see Table 1-2).
RUBOUT or DELETE	Deletes the last character typed on the current line and, if you press the key repeatedly, deletes contiguous characters to the left. Most CRT terminals remove each deleted character from the screen and move the printing position back one space. However, a hardcopy terminal prints a backslash (\), then each character as you delete it, 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. In other words, the first tab stop is character (column) 9.

These control keys will not always work as described here when you use them with various system programs. For example, some editing programs do not read CTRL/Z as an exit. Check the documentation for the system program you are using to determine its use of control keys.

Table 1-2 describes the most commonly used control characters.

Table 1-2 Terminal Control Characters

Character	Description
CTRL/C	Gains the attention of the Monitor Console Routine (MCR), which interprets commands to the operating system. In most cases, the system responds to CTRL/C by displaying the explicit MCR prompt: MCR> This prompt indicates that the system is ready to accept input from your terminal.
CTRL/O	Alternately suppresses and resumes the display of output at your terminal. For example, if you are running a program that generates unwanted output, type CTRL/O. The system then temporarily stops displaying output until you type CTRL/O again to resume output display. (If you do not type another CTRL/O, the system will discard the entire output and then return the default MCR prompt (>) to let you know output is finished.
CTRL/S and CTRL/Q	CTRL/S stops the display of output at your terminal until you type CTRL/Q to resume it. For example, if you are using a CRT terminal that displays output too quickly for you to read, type CTRL/S to halt the display. When you have read what is on the screen, type CTRL/Q to restart the output display. Repeat the process until you have read the entire file.
CTRL/R	Performs a carriage return and reprints the current line on your terminal, omitting any deleted characters, and making the line easier to read. Before terminating the line with a carriage return, type CTRL/R to ensure that you have made the right corrections. For example: MISTKAE\EAK\AKE^R MISTAKE

Table 1-2 (Cont.) Terminal Control Characters

Character	Description
CTRL/U	Deletes the current line. This allows you to retype an entire line when individual corrections would be impractical. Remember that CTRL/U must be typed before the carriage return in order to delete that line. After the carriage return is entered, you must use an editing command from an editor to delete a line.
CTRL/X	Clears your terminal's typeahead buffer and enables you to type additional characters. The typeahead buffer is an RSX-11M feature that allows your terminal to save up to 36 characters before it processes their commands. A "bell character" echoes on your terminal no matter what character you type when this buffer is full. CTRL/X clears the buffer and enables you to resume typing.
CTRL/Z	Exits from many RSX-11M system (and user) tasks and returns control of the system to MCR.

Note the difference between the CTRL/O-CTRL/O pair of commands and the CTRL/S-CTRL/Q commands:

- When you type CTRL/O after a previous CTRL/O, output display resumes further down in the file than the point at which you halted it with the first CTRL/O.
- When you type CTRL/Q after a CTRL/S, output display begins where it stopped. The total output can still be displayed on your terminal.

1.2 MCR COMMANDS

The commands that interpret terminal input to control system operations are called Monitor Console Routine (MCR) commands. You communicate with MCR by entering a command line in the following form:

commandname parameter(s)/keyword(s) line terminator

commandname

Consists of three or more letters, terminated by a space, that uniquely identify an MCR function. MCR only reads the first three letters. Additional letters merely help you identify the command. (The only exception to this 3-character rule is the HELP command. You must

type this entire command to distinguish it from the MCR HELLO command.) Some examples of MCR commands are given below:

Command	Function
ABO[RT]	Stops a running program
DMO[UNT]	Dismounts a volume
UFD	Creates a User File Directory

parameter

Specifies the object of the command function, which is usually a task or device. (A task is the basic executable program on an RSX-11M system.) One or more blank spaces must separate the parameter from the command name, or one parameter from another. For example, when you issue the ABORT command, you include the name of the running task to be aborted. The following command runs the task ADDTWO at 12:25:00

```
RUN ADDTWO 12:25:00
```

The following command aborts the task ADDTWO:

```
ABO ADDTWO
```

/keyword

Modifies either the actual function of the command or a parameter of the command. When a keyword modifies a command function, you must insert a space between the command name and the keyword. For example:

```
commandname /keyword [=value]
```

When a keyword modifies a parameter of a command, the keyword immediately follows the parameter. In addition, some keywords are modified by an added numerical argument, shown in the examples as value. For example:

```
parameter/keyword[=value]
```

Spaces are also not required between consecutive keywords:

```
parameter/keyword/keyword. . .
commandname /keyword/keyword. . .
```

For example:

```
SET /CRT=TI:
```

This command tells the RSX-11M system that your terminal is a CRT. TI: is a pseudo device name that stands for the terminal you are using, that is the terminal from which a command was entered. In this case, CRT is the keyword that determines the function of the SET command.

line terminator

Represents either the carriage return <CR> or return key, or the altmode or escape key (<ESC>). Because <ESC> has a special meaning under certain circumstances, this manual uses <CR> as a terminator. Any terminator causes the line to be sent to the computer or to the task waiting for input.

Before typing a command, you must ensure that the terminal is in the right state to accept commands by:

- Checking that the terminal's power is on
- Checking that the LOCAL/REMOTE switch is set to REMOTE
- Consulting installation instructions for additional required terminal settings or instructions for using a dial-up line
- Typing CTRL/C to obtain the explicit MCR prompt (MCR>).

Both the explicit MCR prompt and the default prompt (>) indicate that MCR is ready to accept input from your terminal. If you enter CTRL/C when you are using a system program other than MCR, you can issue one MCR command. The system executes that command and returns you to the task you were in when you entered CTRL/C. Both DIGITAL-supplied and user-written tasks can request input by displaying a task prompt at a terminal.

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. Terminals connected to a nonmultiuser system are available to any user. Therefore, you can ignore the material in the remainder of this chapter.

1.2.1 The HELP Command

HELP is the only command (besides HELLO, the logging-on command) that you can issue from a terminal before you log on the system. Because the first three letters of HELP are the same as HELLO, you must enter the HELP command in full. When you issue this command, your terminal displays text (determined by each installation) that 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 available to you from the system.

1.2.2 The HELLO Command

To begin a session at the terminal, you must issue the MCR HELLO command to log on.

Logging on the system accomplishes several things:

- It allows the system to make sure that you are an authorized user and to record information about your use of the system
- It grants you the use of the terminal for access to the system until you log off
- It establishes initial default values for your terminal activities

The MCR HELLO command prompts for its parameters as follows:

```
>HELLO<CR>
ACCOUNT OR NAME: [g,m]<CR>
                  or
              last-name<CR>
PASSWORD: password<CR>
```

[g,m]

Commonly called a UIC and 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 member number within the group. The general UIC format consists of the two numbers separated by a comma and enclosed in square brackets [g,m]. When you receive your UIC on a multiuser protection system, a User File Directory (UFD) with the same number is created for you.

Although you have access to other UFDs on the system, your UIC remains the same, no matter which UFD you are working with.

last-name

Your last name, which you can enter instead of a UIC. The system determines your correct UIC from the name you specify.

password

A 1- to 6-character alphanumeric string. The system maintains account information, including correct passwords for each UIC and last name. You cannot gain access to the system unless you type the password corresponding to the UIC or name you have entered.

The system does not print (echo) on your terminal the characters you type in response to the PASSWORD prompt. This ensures that your password remains private and that no one can log on using your account.

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

An example of the log-on procedure follows:

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

   RSX-11M BL26 MULTI-USER SYSTEM

GOOD AFTERNOON
12-JAN-79  11:07  LOGGED ON TERMINAL TT32:

10-JAN-79

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

>
```

When the system receives the correct password the terminal displays a message that includes system identification and the date and time you logged on the terminal. Notice the password is not echoed in the example above; this is the way your terminal will look when you enter your password. The terminal can display additional messages, which usually supply information that affects general use of the system. The default MCR prompt (>) on the line after the messages indicates that the messages are complete and MCR is ready to accept commands.

If the following message appears when you try to log on, someone else is using the terminal:

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

Either try to log on a different terminal or find the person who logged on before you. Do not use the terminal without the other user's knowledge.

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 log-on procedure.

Suppressing System Messages

The HELLO command allows you to enter a special form of the UIC parameter to suppress the display of most of the system messages. (The system message file text can be written so that important messages cannot

be suppressed.) The normal format for a UIC is [g,m]. However, if you do not want to see the system messages, replace the comma with a slash (/) as follows:

[g/m]

HELLO allows four UIC formats:

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

Either of the slash formats suppresses all but the most important of the system messages. When you log on, brackets around the UIC are optional. However, a UIC must always be specified in the format [g,m] when it is a parameter of any other command.

Suppressing ACCOUNT OR NAME Prompt

Another way to shorten the time required to log on is to enter the HELLO command and UIC (or last name) on the same line using a slash character as a separator:

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

This command format suppresses the prompt for ACCOUNT OR NAME.

Automatic Execution of Log-On Command File

After you log on a terminal, the system automatically assigns the name SY: to your system disk and then searches your UFD on SY: for the file LOGIN.CMD. If found, the system sends LOGIN.CMD to the indirect command processor, which executes all of the commands in LOGIN.CMD.

This feature is useful when there are specific commands you issue each time you log on the system.

For example, you could use your login command to set your terminal to be a CRT and to accept lower case letters. You could also include in the file a command to print a text file containing messages to yourself (just as the system prints a text file containing messages to all users when you log on). When you want to change the messages to yourself, you simply edit the text file MYLOGIN.TXT below. In this case, the following message is displayed on your terminal after the system messages:

```
>@LOGIN.CMD
>SET /LOWER=TI:
>SET /CRT=TI:
>PIP TI:=MYLOG.TXT
10:00       MEETING WITH MANAGERS
11:00       RACQUETBALL WITH JACK
12:30       FINANCE COMMITTEE MEETING
>@<EOF>
```

1.3 ERROR MESSAGES

When MCR receives input that it does not recognize or that it knows to be incorrect, it displays a message on your terminal. An example is the message that appears if you try to log on a terminal already in use. You will receive a different message from MCR if you enter a UIC (or last name) or password that the system does not recognize:

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

This attempt to log on was not successful, either because the system did not recognize the last name (CONRAK), or because you did not enter the correct password.

The appropriate user response to an error condition depends on the message displayed. All messages returned by MCR commands are explained in the *RSX-11M/M-PLUS MCR Operations Manual.* Command descriptions in that manual include a list of possible messages that the command can generate. The manual also includes an alphabetical list of all MCR messages.

Each message a task prints on your terminal begins with the 3-letter name of the associated command or task sending you the message. When you encounter an error while running a system task, such as a text editor, look for an explanation of the message in the documentation for that task.

CHAPTER 2
PROGRAM PREPARATION

Four steps are required to prepare a program to run on RSX-11M:

1. Create a source file
2. Compile or assemble the source file to produce an object module
3. Link the object module to create a task image file
4. Run the task (using an MCR RUN command)

Each of the steps involves the manipulation of at least one file, which is an owner-named and created area on a mass storage volume (usually a disk or tape).

When you log on a terminal, the system automatically allows you access to a disk. This disk is your default system disk on which RSX-11M stores all your files unless you specify otherwise. (If your system does not have multiuser protection, you have automatic access to a system disk that is also accessible to all users.) This chapter assumes you are using your default system disk.

Note that the RSX-11M operating system lets you run programs written in a number of computer languages. The MACRO-11 assembly language is always included with RSX-11M. Check with your system manager to see what other languages are available to you.

Each language has its own system program (called an assembler or compiler, depending on the language). This system program translates a source program into an object module and makes sure that the source program follows the language syntax rules.

Each language compiler or assembler can only translate source files written for that language: the FORTRAN-IV program in this book must be compiled by a FORTRAN-IV compiler (see specific language documentation for more information.)

Although this manual illustrates the creation of a FORTRAN-IV program, you follow the same steps to create a MACRO program, a COBOL program, or a program in any other compiled or assembled language. The only difference is in the language translator you use.

2.1 CREATING THE SOURCE FILE (EDI)

The RSX-11M Line Text Editor (EDI) is a system program you can use to create a source file. (You can also use EDI to create other types of files, such as text files and data files.) RSX-11M includes two other editing programs: the DEC Standard Editor (EDT) and the Source Language Input

Program (SLP). EDT is especially useful with a CRT terminal because it lets you edit by character as well as by line. SLP is used primarily to maintain system programs. (See the *RSX-11 Utilities Manual* for documentation of EDT and SLP, as well as additional information about EDI.)

This chapter uses a limited number of EDI commands to show you how to create and edit the source file for ADD.FTN.

2.1.1 Using EDI to Create a Source File

To invoke EDI, issue a call to the editor in the same way that you issue an MCR command. (Remember you can always type CTRL/C to get the explicit monitor prompt and make sure your command is going to MCR.)

```
>EDI<CR>
```

EDI displays its task prompt:

```
EDI>
```

To create a new file using EDI, specify a file name and a file type in the following form:

```
EDI>filename.filetype
```

filename

A 1- to 9-character alphanumeric string.

filetype

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

Type	Language
.BAS	BASIC
.CBL	COBOL
.FTN	FORTRAN
.F4P	FORTRAN-IV-PLUS
.MAC	MACRO

The following example, which is used throughout this chapter, illustrates the creation of a FORTRAN IV source file called ADD.FTN. ADD.FTN is an interactive program that prompts you to type two numbers at a terminal. The program then adds the two numbers and prints the result on your terminal.

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

When EDI receives the name of a new file, it creates an empty file with that name and type and displays the two lines shown in the example above. EDI prints INPUT to let you know it is ready to accept input from your terminal. Anything you type except for control characters becomes part of the file called ADD.FTN. You can then type in the source program for ADD.FTN. When you terminate each line of input with a carriage return <CR>, EDI stores the line in a buffer. When you end the editing session, EDI writes the entire buffer to the file ADD.FTN.

You can use the keyboard facilities described in Section 1.1 to correct any mistakes on the current line before you type <CR>. Once a line has been terminated and written to the buffer, you must use EDI commands to make any changes.

You can also enter the new file name and file type on the same line as the call to EDI. This quicker way to create a file is illustrated below. Note the use of the DELETE or RUBOUT key, CTRL/R, and CTRL/U in the example:

```
>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^R
          ACCEPT 2,K,L<CR>
2         FORMAT (222\22\I5)<CR>
          PRINT^U
          TYPE 3,K+L<CR>
3         FORMAT (' THE SUM IS ',I5)<CR>
          STOP<CR>
          END<CR>
<CR>
*EX<CR>
[EXIT]
>
```

After you terminate the last line of text in the program, type a carriage return as the first character in the new line. EDI responds by displaying an asterisk (*) prompt. Until now, EDI was in Input Mode, the mode it entered to create a new file. Typing a carriage return (<CR>) at the beginning of a new line switches EDI to Edit Mode. The asterisk is EDI's prompt for commands.

The command EX[IT] instructs EDI to write ADD.FTN to your disk area and to return control of your terminal to MCR.

The KILL command closes both input and output files and deletes the output file. It eliminates all traces of the current editing session.

2.1.2 Editing an Existing File

To edit an existing file with EDI, enter the same command you used to create a new file:

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

Because a copy of ADD.FTN now exists on disk, EDI responds differently, as follows:

```
>EDI ADD.FTN<CR>
[00008 LINES READ IN]
[PAGE 1]
*
```

EDI creates a copy of ADD.FTN and enters Edit Mode, indicated by the asterisk (*) prompt. The message [00008 LINES READ IN] tells you the number of lines EDI has placed in its buffer. The lines in the buffer make up the text currently available for editing. The buffer may or may not contain the entire input file, depending on the size of the file and the buffer. To access text beyond the current buffer, issue a RENEW command in Edit Mode. Renew writes the current buffer to the output file and refills the buffer with the next block of text.

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

Locating and Changing Text

The EDI commands listed below allow you to find and change text in a file. You can abbreviate most EDI commands to one or two letters. In the following text, the optional portion of each command is in square brackets [ ].

Command	Function
L[OCATE]	Locates a string of text in the current buffer
C[HANGE]	Replaces one text string with another
N[EXT]	Advances the line pointer to the next line (EDI displays the Edit Mode prompt (*) but does not display the line)
P[RINT]	Displays the current line on your terminal
T[OP]	Positions the line pointer at the top of the current buffer
BO[TTOM]	Positions the line pointer at the bottom of the current buffer
<CR>	Points to and displays on your terminal the next line in a file

An example of an EDI editing session is shown below with text explaining what each command does to change or to locate lines in the file and print them on the terminal.

```
>EDI ADD.FTN<CR>
[00008 LINES READ IN]
[PAGE 1]
*LOCATE ENTER<CR>
1         FORMAT (' ENTER TWO NUMBERS - M,N')
*
```

The LOCATE command in the example above points to and prints the next line in the file (after the current line) containing the word ENTER.)

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

The CHANGE command above changes the word ENTER to TYPE. Note that the slash characters (/) are used to delimit both the old and the new text strings. Any ASCII characters that are not used in either string can be used to delimit a string. EDI then prints the corrected line on your terminal.

```
*NEXT<CR>
*
```

The NEXT command points to the next line in the text, but does not print it on your terminal.

```
*PRINT<CR>
          ACCEPT 2,K,L
*
```

The PRINT command above displays the current line on your terminal.

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

The LOCATE command points to and prints the next line in the file containing the word SUM.

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

The CHANGE command above changes the word SUM to RESULT.

```
*LOCATE (2I5)<CR>
[*EOB*]
```

The LOCATE command searches for a line containing (2I5). In this case, EDI reaches the end of the buffer (EOB) without finding the string (2I5). The EDI line pointer only moves forward through the buffer in response to a LOCATE command. It does not search backward through a file.

```
*TOP<CR>
*
```

The TOP command moves the line pointer to the top of the buffer (one line before the first line of text). At that point, Edit Mode displays the asterisk prompt (*).

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

When you enter a carriage return in response to the asterisk prompt EDI prints the next line on your terminal. The carriage return also moves the EDI line pointer to the next line. Therefore, it performs the same function as the NEXT and PRINT commands. In this example the pointer points to the first line in the buffer.

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

The EXIT command writes the current buffer and the remainder of the input file to the output file, closes both input and output files, and exits from EDI to MCR.

Inserting and Deleting Text
The EDI commands listed below allow you to insert and delete text in the file.

Command	Function
I[NSERT]	Inserts one or more new lines of text in a file.
A[DD]	Adds text to an existing line.
D[ELETE]	Deletes the current line.
<ESC>	Points to and prints the previous line. Note, however, that <ESC> does not display the line you just entered with an Insert command. It moves the pointer up one line in the text and displays that line.
R[ETYPE]	Replaces the current line of text with a new line.
LI[ST]	Prints on your terminal the lines remaining in the buffer, from the current line to the end. After the LIST command executes, the line pointer is reset to the top of the buffer.
TYPE n	Prints on your terminal the next n lines, but does not reset the line pointer.

An example of an EDI editing session is shown below with text explaining what each command does to insert or delete text.

```
>EDI ADD.FTN<CR>
[00008 LINES READ IN]
[PAGE 1]
*
```

EDI retrieves the latest version of ADD.FTN. Note that if you want to edit an earlier version of ADD.FTN, you can include a version number in this command.

```
*INSERT<CR>
C         THIS PROGRAM ADDS TWO NUMBERS<CR>
<CR>
*<ESC>
[*BOB*]
<CR>
C         THIS PROGRAM ADDS TWO NUMBERS
*
```

The EDI INSERT command in the text above switches to EDI Input Mode and inserts text immediately before the first line of text already in the buffer. The second carriage return takes you back to Edit Mode. The <ESC> command from Edit Mode causes EDI to display the line before the current line. In this case, the line before the current line is the beginning of the buffer [*BOB*]. The carriage return moves the pointer to the line you just inserted, which is now the first line of the text.

```
*LOCATE NUMBERS<CR>
1         FORMAT (' TYPE TWO NUMBERS - M,N')
*ADD !PROMPT FOR INPUT<CR>
*PRINT<CR>
1         FORMAT (' TYPE TWO NUMBERS - M,N')!PROMPT FOR INPUT
*
```

The LOCATE command finds the line on which a string occurs, the ADD (append) command attaches a comment text to the line, and the PRINT command displays the revised line.

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

The DELETE command deletes the current line and moves the pointer to the next line.

```
*<ESC>
          TYPE 3,K+L
*
```

The <ESC> command causes EDI to point to and print the previous line in the buffer on your terminal.

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

EDI enters insert mode and inserts a new line of text. The second carriage return causes EDI to reenter Edit Mode at the current line.

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

The TOP command repositions the line pointer to the top of the buffer. The carriage return points to and prints the next line, which in this case is the first line in the buffer. The RETYPE command replaces the current line with the text that follows the command.

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

The LIST command displays the lines from the current line to the end of the buffer.

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

When you issue the EXIT command, EDI writes the file to your disk area and returns control to MCR.

2.1.3 Basic EDI Commands

Table 2-1 summarizes the basic set of EDI commands. These commands provide all the functions you need for simple editing.

Table 2-1 Basic EDI Commands

Command	Description
A[DD] string	Appends string to the current line.
BO[TTOM]	Moves the line pointer to the bottom of the current buffer.
C[HANGE]/string1/string2/	Replaces string1 with string2 in the current line.
<CR>	When in Input Mode, <CR> returns you to edit mode. When in Edit Mode, <CR> prints the next line on your terminal and moves the line pointer to that line.
CTRL/Z	Closes the input and output files and terminates the editing session.
D[ELETE]	Deletes the current line.
<ESC>	Points to and prints the previous line.
EX[IT]	Closes the input and output files and terminates the editing session.
I[NSERT] string	Inserts string before the next line or enters input mode if string is omitted.
KILL	Closes both input and output files and deletes the output file.
L[OCATE] string	Locates the next line containing the object string. The search stops at the end of the current buffer.
N[EXT]	Advances the line pointer to the next line.
P[RINT]	Prints the current line on your terminal.
REN[EW]	Writes the current buffer to the output file and reads in another buffer from the input file.

Table 2-1 (Cont.) Basic EDI Commands

Command	Description
R[ETYPE] string	Replaces the current line with the object string.
T[OP]	Moves the line pointer to the top of the buffer.
TYPE	Prints on your terminal the next n lines but does not reset the line pointer.

2.2 COMPILING A FORTRAN-IV SOURCE FILE

After you have created a FORTRAN-IV source file, in this case ADD.FTN, the next step is to compile the program into machine-readable (binary) form. This compiled program is called an object module.

2.2.1 Creating an Object Module

You begin the compilation process by issuing a call to the FORTRAN-IV compiler (in the same way you issued a call to the editor, EDI):

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

This command line tells the FORTRAN-IV compiler to compile the source file ADD.FTN to produce an object module called ADD.OBJ. As the example shows, you specify from left to right the output file (ADD.-OBJ), an equal sign (=), and the input file (ADD.FTN).

NOTE

RSX-11M command line specifications follow the convention used by many computer languages: the output of a process appears to the left of the equal (=) sign and the input to the process appears to the right. (See Chapter 3 for more information on file specifications.)

The FORTRAN-IV compiler defaults the object file type to .OBJ and the source file type to .FTN. This means that if you do not include a file type with the file name in a command to the compiler, FORTRAN searches for a file with the name you supply and a file type of .FTN. If it finds one, it uses that file as input. FORTRAN creates a file with a type of .OBJ as output if you do not specify an output file type.

Most RSX-11M system programs and all language processors use defaults in this way. For this reason, you should use the standard file types whenever possible. (Additional standard file types are described in the *RSX-11M/M-PLUS MCR Operations Manual* and are listed in the *RSX-11M Mini Reference.)*

The default file types in the FORTRAN-IV command line are shown in Table 2-2.

Table 2-2 Default FORTRAN-IV Compiler File Types

File		Default File Type
Input	**Output**	
Source File		.FTN
	Listing File	.LST
	Object Module	.OBJ

Using the defaults, you can shorten the above command to:

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

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

2.2.2 Creating a Listing File

In addition to translating a source program to an object module, the FORTRAN-IV compiler can also supply a listing file and a storage map for the program. To create a listing file you enter the command:

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

The second ADD on the output side of the command instructs the compiler to produce a listing file and storage map (called ADD.LST) for the source program. The listing file provides information on compiler errors and the storage map lists the symbolic names used in the program and flags references to symbols that will have to be resolved by the Task Builder.

If the system includes the Queue Manager and spooling to a line printer, the FORTRAN-IV compiler automatically puts ADD.LST in the line printer queue. You can inhibit this automatic printing with the following command line:

```
>FOR ADD,ADD/-SP=ADD<CR>
```

The -SP switch tells the compiler not to spool the listing.

The compiler also stores a copy of ADD.LST on your system disk. (If your system does not include spooling, the compiler only stores the file on disk.)

The FORTRAN compiler can also produce a listing file of your program without creating an object file. This feature enables you to find errors in your program using the listing file, without generating multiple versions of unusable object files. To produce only a listing file, enter a command in the following form:

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

Following this command, ADD.LST is both spooled to the line printer on systems with spooling and also stored on your disk area.

Note that you must enter a comma before the listing file specification in the command line; otherwise the compiler interprets ADD as an object module instead of a listing file.

The following command causes a copy of ADD.LST to be displayed at your terminal but not printed on the line printer or saved on disk:

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

2.2.3 Requesting a Nonresident FORTRAN-IV Compiler

In some cases, you may issue the FOR command and receive the message:

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

This means that the FORTRAN compiler is not currently installed on your system. To install the compiler, issue the MCR command:

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

This command tells the system to install the FORTRAN compiler in memory and to activate it at your terminal. The compiler responds with the prompt:

```
FOR>
```

You then enter the FORTRAN-IV command as illustrated previously:

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

After the compiler executes your commands, it displays the task prompt (FOR>) once again. At this point, you can compile additional FORTRAN programs or exit from the compiler. Enter CTRL/Z to terminate the compiler and remove it from memory. (See the *RSX-11M/M-PLUS MCR Operations Manual* for details on the RUN command.)

2.3 BUILDING A TASK IMAGE

The RSX-11M Task Builder converts the object modules created by a language translator into a single task image file. This task image file resides on disk until someone issues an MCR command to install and run it. (You may have used computer systems which referred to some parts of the Task Building process as linking or loading.)

2.3.1 Full Task Build Command Line

A full Task Builder command line uses three output files and any number of input files. The output files are:

1. The task image file (filename.TSK) which contains the task image to be installed and run
2. A memory allocation file (filename.MAP) which contains information about the size and location of various parts of the task
3. A symbol definition file (filename.STB) which contains information about the task's global symbol definitions (The STB file is not needed or created in most Task Builder operations.)

The input files include all of the object modules and modules from system or user libraries required to produce a single task image. The full Task Builder command line is:

```
>TKB task image file, map file, symbol definition file=object file(s)
```

Table 2-3 illustrates the default file types in the Task Builder command line.

Table 2-3 Default Task Builder File Types

File		Default File Types
Input	**Output**	
Object File		.OBJ
	Task Image File	.TSK
	Task Builder Map	.MAP
	Symbol Definition	.STB

To Task Build ADD.FTN, issue the following command:

```
>TKB ADD,TSK,ADD,MAP,ADD,STB=ADD,OBJ,LB:[1,1]FOROTS/LB<CR>
```

In this command, LB: is a pseudo device name for the device containing the library, [1,1] is the UFD where the library is stored, and FOROTS is the FORTRAN Object Time System library. The *RSX-11M/M-PLUS Task Builder Manual* describes the use of the LB switch to include OTS libraries in task image files.

Using Task Builder defaults, the command can be shortened to:

```
>TKB ADD,ADD,ADD=ADD,LB:[1,1]FOROTS/LB<CR>
```

2.3.2 Short Task Build Command Lines

Like the compiler, the Task Builder allows you to omit any output file or to direct output files to your terminal or to a line printer. However, if you omit an output file in the beginning or middle of the output side of a file specification, you must retain the comma in place of the file name to specify a null (empty) field. On the input side of the command, use a comma to separate one input file from another. Table 2-4 illustrates the creation of Task Builder output files.

Table 2-4 Task Builder Output Files

Command Line	Output Files Generated
>TKB .TSK,.MAP,.STB=.OBJ	All three output files
>TKB ,,.STB=.OBJ	Symbol definition file only
>TKB ,.MAP,.STB=.OBJ	Memory allocation and symbol definition files
>TKB .TSK,,.STB=.OBJ	Task image and symbol definition file
>TKB .TSK,.MAP=.OBJ	Task image and memory allocation files
>TKB .TSK=.OBJ	Task image file only

2.3.3 Multiline Task Build Command Procedures

A long list of input files can cause a Task Builder command line to exceed the terminal's maximum line length. If this happens, you can invoke the Task Builder in the following manner:

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

After you enter the carriage return, MCR activates the Task Builder, which then returns the TKB> prompt. The Task Builder displays its prompt after each line of input until you enter two slash characters (//) and a carriage return at the beginning of a line. The slash characters terminate operation of the Task Builder and return you to MCR. For example, the single command line:

```
>TKB
TKB>filnam.TSK,filnam.MAP,filnam.STB=filnam.OBJ,OBJ,OBJ<CR>
```

can be entered as:

```
TKB>filnam.TSK,filnam.MAP,filnam.STB=filnam1.OBJ<CR>
TKB>filnam2.OBJ,filnam3.OBJ<CR>
TKB>//<CR>
>
```

2.3.4 Listing the Memory Allocation (MAP) File

The memory allocation file (map file) is an ASCII file that contains information about the size and location of components within the task. If your system includes the queue manager and print spooling options, the Task Builder spools the file to the line printer by default.

Without the spooling option, however, the Task Builder only stores the file on your system disk.

2.3.5 Task Builder Switches and Options

The Task Builder includes a variety of switches and options to control the creation of a task image on disk. These switches and options are described in detail in the *RSX-11M/M-PLUS Task Builder Manual.*

2.4 RUNNING A TASK

To run the task you created, issue an MCR RUN command as follows:

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

The RUN command instructs the system to:

- Locate ADD.TSK on your default system disk
- Load a copy of the task image into memory
- Execute the task

When you use this form of the RUN command, the task image file remains on disk, ready to be run again, until it is explicitly deleted.

The following sections illustrate three successful executions of the FORTRAN-IV program ADD.TSK. Remember that ADD is an interactive program; that is, it prints a message on your terminal and then waits for you to type in a response before it performs the calculations and prints the result on your terminal.

```
>RUN ADD<CR>
TYPE TWO NUMBERS - M,N
7,3<CR>
THE SUM IS 10
TT47 -- STOP
>
```

```
>RUN ADD<CR>
TYPE TWO NUMBERS - M,N
522,628<CR>
THE SUM IS 1150
TT47 -- STOP
>
```

```
>RUN ADD<CR>
TYPE TWO NUMBERS - M,N
9,16<CR>
THE SUM IS 25
TT47 -- STOP
>
```

Note that when the program completes execution, it displays the line:

```
TT47 -- STOP
```

TT47 is a temporary name assigned to the task when you issue the RUN command. The version of the RUN command used in this example causes the system to name the task TTn, where n is the unit number of the terminal running the program.

Figure 2-1 reviews the steps to creating the FORTRAN-IV task ADD, including the specific files created in each step.

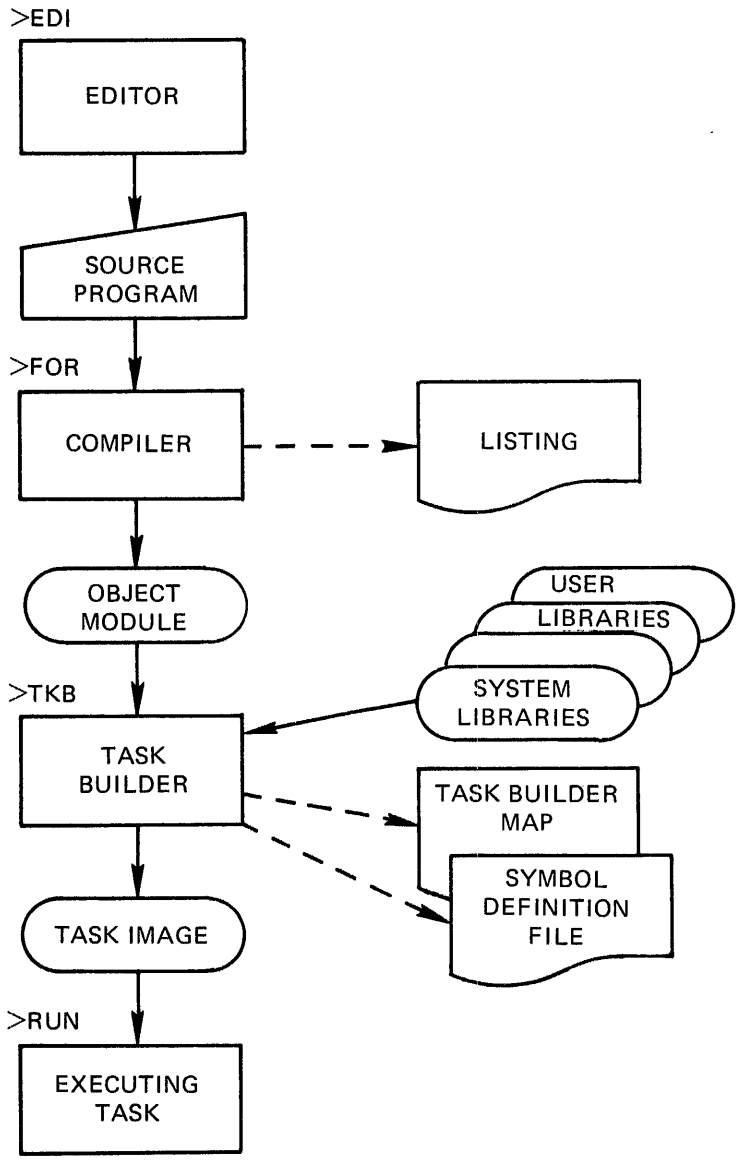

Figure 2-1 Development of a FORTRAN-IV Task

CHAPTER 3
THE FILES

The RSX-11M file specification is used by all RSX-11M system programs, including editors, compilers, utilities, and the Task Builder.

The examples in Chapters 1 and 2 emphasized using defaults in command line file specifications. Defaults are useful because they allow you to use the resources of the system without understanding every possible choice available to you. They also permit you to do less typing. In many situations, however, you will need a complete file specification, because defaults alone will not cause the system to do what you want it to do.

The format of a full file specification is:

```
dev:[g,m]filename.filetype;fileversion
```

dev:	The name of the device that holds the volume on which the input file resides (or on which the output file will reside).
[g,m]	A User Identification Code (UIC) identifying the User File Directory (UFD) that contains (or will contain) the file.
filename	The 0-9 character name you supply for the file.
filetype	The 0-3 character type you supply for the file.
fileversion	An octal number that distinguishes between different versions of the same file.

Examples of full file specifications are:

```
MT1:[116,23]CHARLA.FOO;32
DM1:[203,204]FOO.FUM;5
DK2:[34,63]WHO.FOO;3
```

Chapter 2 describes the file name and file type fields of a file specification. The following sections describe the device name and UFD fields.

3.1 THE DEVICE NAME

The device name specifies the volume on which the file resides. The name consists of two alphabetic characters and an optional 1- or 2-digit octal unit number followed by a colon (:). When the name does not include a unit number, the system tries to use unit number 0.

Device names in the above examples and their corresponding abbreviations are listed below.

Name	Physical Unit
DK2:	RK05 disk, unit 2
DM1:	RK06 disk, unit 1
MT1:	TU10, TE10, or TS03 magnetic tape, unit 1

The device name can refer to one of two kinds of devices:

- An actual physical device, such as the three listed above

  or

- A pseudo device which can represent a variety of physical units, depending upon which terminal enters the unit name

For example, the name TI: is a pseudo device that refers to the terminal from which input is being entered. When you enter input from terminal 23 on your system, your physical terminal number is TT23: and your pseudo terminal is TI:. When you enter input from terminal 6, your physical terminal number is still TT6: but your pseudo terminal remains TI:.

Therefore, you can either write a program to send output to TT23:, which means you will always have to run the program from TT23:, or you can write it to send output to TI:, which means it will always send output to whatever terminal installs and runs the task.

Another pseudo device name is SY:, which corresponds to your default system disk. All the files created so far in this manual reside on SY:.

Section 2.3.1 describes the pseudo device LB:, the volume that contains libraries.

Table 3-1 lists commonly used devices. The letter n stands for the unit number of the device. There are different types of disks and magnetic tapes, which have correspondingly different device names. Ask someone familiar with your system which names you should use. (See the *RSX-11M/M-PLUS MCR Operations Manual* for a complete list of device names.)

Table 3-1. Device Names and Abbreviations

Device	Abbreviation
DECtape	DDN: DTn:
Disk	DBn: DFn: DKn: DLn: DMn: DPn: DRn: DSn: DXn: DYn:
Line Printer	LP:
Magnetic Tape	MMn: MSn: MTn:
Pseudo Terminal	TI:
Terminal	TTn:
Pseudo System Device	SY:

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

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

3.2 USER FILE DIRECTORIES

When you log on, you either use your login UIC itself or specify it indirectly by using your last name. This UIC identifies the default User File Directory (UFD) set up for you on your system disk (SY:) when the system manager made you an authorized user. This UFD is itself a file that lists the names of all files stored in your directory. None of the system programs can locate a file unless they know the UFD in which the file is listed.

The terms UIC and UFD are often used interchangeably in RSX-11M. The UIC, however, identifies the user and the UFD identifies the directory.

When you need to use files stored in other directories, you can use an MCR SET command to change your default UFD (SET /UIC=[g,m]) or you can specify a UFC in the file specification. The MCR SET command changes the default UFD. However, neither of these actions changes the UIC under which you logged on.

For example, if you want a copy at your terminal of a file from another UFD, issue the command:

```
>PIP TI:=[302,200]CONRAD.MAC<CR>
```

This command assumes that CONRAD.MAC resides on your default system disk, in UFD [302,200].

In a system without multiuser protection, your default UFD corresponds to the UIC specified in the last SET /UIC command issued from the terminal you are using.

To display the current default UIC on any system, issue the command:

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

When you issue a set command in this form, without specifying what UIC is to be set, the system responds with:

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

CHAPTER 4
PIP AND THE QUEUE MANAGER

The following sections introduce two utilities, the Peripheral Interchange Program (PIP) and the Queue Manager, and show you how to use these utilities to copy files, list the files in a directory, delete, rename, and purge files, and print files on a line printer or display them on your terminal.

4.1 PERIPHERAL INTERCHANGE PROGRAM

The Peripheral Interchange Program (PIP) manipulates files on RSX-11M by using appropriate switches to perform the operations.

Among other things, PIP allows you to

- Copy files from one UFD or device to another
- Delete files
- Rename files
- List directories of files
- Purge files

Any volume you refer to in a file specification must either be allocated as public (available to all system users) or be allocated to you (on multiuser protection systems only) and mounted. If any volume specified or implied in a PIP command is not mounted or is allocated to another user, PIP returns the message:

```
PIP -- DEVICE NOT MOUNTED/ALLOCATED
    dd:[s,m]
```

Device allocation, mounting, and UFD creation are all MCR functions accomplished by the following MCR commands:

- ALLOCATE
- MOUNT
- UFD

(See the *RSX-11M/M-PLUS MCR Operations Manual* for a description of these commands.) In the process of creating, editing, compiling and Task Building ADD.FTN, RSX-11M created a number of files on disk:

- 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 repeated any of the program preparation steps

All of these files remain on disk taking up space until you explicitly delete them. RSX-11M does not automatically eliminate old versions of files. (Appendix A includes an example of how consecutive operations can create multiple copies of files.)

You can invoke PIP from MCR in either of two ways:

1. The single line format, which executes one PIP command and returns control to MCR:

```
>PIP /LI<CR>

DIRECTORY DB0:[301,314]
14-FEB-79  09:10

ADD.FTN;3          1.        22-FEB-79 11:40
ADD.FTN;2          2.        22-FEB-79 12:00
ADD.FTN;1          1.        19-FEB-79 11:32
```

   or

2. The format that passes control to PIP and allows you to execute multiple PIP commands:

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

DIRECTORY DB0:[301,314]
14-FEB-79  09:10

ADD.FTN;3          1.        22-FEB-79 11:40
ADD.FTN;2          2.        22-FEB-79 12:00
ADD.FTN;1          1.        19-FEB-79 11:32

TOTAL OF 4./11. BLOCKS IN 3. FILES

PIP> ADD.FTN /PU<CR>
PIP>^Z
>
```

This chapter uses the one-line form shown in number 1 to illustrate PIP command operations. Any command shown here can also be executed in the method described in number 2.

4.2 PIP COMMAND LINE AND DEFAULTS

The format of PIP command lines differs for each function. In general, however, PIP switches operate on lists of file specifications. PIP uses the last value it encounters in a command line as the default value for that command.

You can enter multiple file specifications in a single command line by separating the individual file specifications with commas. The switch follows the last file specification.

In the following example, PIP will delete the files named ARRAY.FTN;1, MULT.FTN;1 and HEX.FTN;1:

```
>PIP ARRAY.FTN;1,MULT.FTN;1,HEX.FTN;1 /DE<CR>
```

The individual file specifications within the list must each meet the requirements of the PIP switch involved. Therefore, all of the files being deleted in the example must include version numbers because the PIP Delete switch requires version numbers.

Table 4-1 summarizes the PIP switches which are described in this chapter. (See the *RSX-11 Utilities Manual* for additional information on PIP.)

The following sections describe and illustrate the use of these commands.

4.2.1 Displaying a File on Your Terminal

To display a copy of ADD.FTN on your terminal, enter a PIP command in the following format:

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

This command produces the following display:

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

Table 4-1 PIP Switches and Command Functions

Switch	Meaning	Description
/LI	List	Prints the names of requested files contained in the specified UFD.
/BR	Brief	Prints a shortened version of the directory generated by the list command.
/DE	Delete	Deletes the files specified.
/PU	Purge	Purges (deletes all but the highest numbered version(s)) of the files specified.
/SP	Spool	Spools (lists on the line printer) all files specified. (Note that you can also use the Print command of the Queue Manager for this function.)
/RE	Rename	Renames the input file(s).
/CO	Contiguous	Copies the file into contiguous blocks on the output volume.
/TR	Truncate	Truncates unused blocks at the end of the file.
/CD	Creation Date	Copies the file, using the original creation date, instead of updating it to the current date.
/SD	Selective Delete	Deletes the files specified, after asking you whether you want to delete each particular file.

4.2.2 /LI – Displaying Your User File Directory

All of the files you create are listed in your UFD. The PIP /LI command displays the UFD as follows:

```
>PIP /LI<CR>  (1)
DIRECTORY   DB0:[210,76]
10-FEB-79   14:48
   (3)  (2)          (4)      (5)          (6)
ADD.OBJ;2              2.               25-JAN-79 14:31
ADD.TSK;1             19.      C        20-JAN-79 14:32
ADD.MAP;1              4.      C        25-JAN-79 14:32
ADD.STB;1              3.               12-JAN-79 10:37
ADD.FTN;1              1.               10-JAN-79  8:16
ADD.TSK;3             29.      C         8-JAN-79 12:01
ADD.OBJ;1              2.               29-DEC-78 13:06
ADD.TSK;2             29.      C        29-DEC-78 13:04

TOTAL OF 99./112. BLOCKS IN 8 FILES
            (7)
>
```

The PIP /LI version of the command includes the following information, keyed to the numbers in the directory above:

(1) The physical volume on which the files are stored, and the UIC that owns the directory. The volume named is your default system disk (SY:). The UIC is your default UFD.

(2) The date and time PIP created the directory listing.

(3) The name, type, and version number of each file.

(4) The number of blocks used by each file. A block is 512 bytes (256 words) long.

(5) A file code which indicates the file is either contiguous (C code), noncontiguous (no code), or locked (L). Locked files usually contain corrupted data. You can access them by using the PIP Unlock switch (PIP filename/UN).

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

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

PIP also provides switches to control the amount of detail given in the directory listings. The BR switch causes PIP to generate a directory in brief form. For example, the command PIP /BR causes PIP to print a directory on your terminal in the form:

```
PIP /BR<CR>

DIRECTORY DB1:[303,12]

ADD.FTN;3
ADD.FTN;2
ADD.FTN;1
```

The FU switch causes PIP to print a directory in more detailed (full) form, as shown in the following example:

```
PIP /FU<CR>

DIRECTORY DB1:[303,12]
21-FEB-79 14:49
                          (1)                 (2)
ADD.FTN;3          (1437,27)          73./73.          20-FEB-79 14:23
                          (3)                 (4)
  [303,12]  [RWED,RWED,RWED,R]          21-FEB-79 14:07(4.)
ADD.FTN;2          (6206,22)          57./57.          20-FEB-79 13:41
  [303,12]  [RWED,RWED,RWED,R]          21-FEB-79 14:22(4.)
ADD.OBJ;1          (21142,3)          66./66.          20-FEB-79 13.19
  [303,12]  [RWED,RWED,RWED,R]          21-FEB-79 14:22(4.)
```

This form provides the following information (in addition to the information explained in the preceding example):

(1) A file identification number consisting of the file number and the file sequence number. This information is used by the RSX-11M file system to keep track of your files.

(2) Blocks allocated to each file (in addition to the blocks used).

(3) A protection code which tells how your file can be accessed by other system users.

(4) The date and time the file was last revised and the number of revisions.

After displaying the directory in any of these forms, PIP returns control to MCR.

4.2.3 Display Information on Specific Files

PIP allows you to obtain information about one file or a specific group of files in a directory. 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 of all versions of the file called ADD.TSK. The asterisk (*) in the version number field tells PIP to search for all versions of the file.

The asterisk (*) character in one or more fields of a file specification stands for all. This is also called a wildcard specification. PIP restricts the use of wildcards in the cases listed below.

The following restrictions apply to PIP output file specifications:

1. Copying a single file
2. Listing a directory
3. When you copy several files, the output file specification must be *.*;* or default.
4. The output specification, used with the Rename and Enter switches, can mix wildcards with specified fields. With either switch, PIP uses the equivalent field of the input field specification.

In all cases where PIP permits wildcards in the output file specification, the wildcard UIC form [*,*] is used to indicate that the output UIC is the same as the input UIC.

The following restrictions apply to wildcards on PIP functions with input files:

1. *.*;* means all versions of all files in the current UFD
2. *.DAT;* means all versions of all files of file type DAT in the current UFD
3. TEST.*;* means all versions and all types of files with the file name of TEST in the current UFD
4. *.* means the most recent version of all files in the current UFD
5. *.DAT means the most recent version of all files of file type .DAT in the current UFD
6. TEST.* means the most recent version of all file types for files named TEST in the current UFD
7. TEST.DAT means the most recent version of TEST.DAT in the current UFD
8. [*,*] means all UIC group, member combinations from 1 to 377
9. [n1,*] means all UIC member numbers under group n1
10. [*,n2] means all group numbers for member n2

ADD.*;* refers to all versions of all types of the file named ADD in the current UFD.

4.2.4 Deleting Files

Once you know what files are listed in your UFD, you can decide which files you want to delete. For example, to retain the highest numbered version of each of the files listed in Section 4.2.2, you would delete the following files:

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

To delete these files with the PIP Delete switch, issue the following command:

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

Note that the Delete switch requires either an explicit version number or a wildcard in the version field. However, the use of a wildcard is inappropriate when your UFD includes files with the same name but other types and versions that you do not want to delete. In this instance, you must specify explicitly each file you want to delete.

4.2.5 Purging Files

When you want to eliminate all but the highest version of files, the PIP Purge switch is often more efficient than the Delete switch. The following Purge command has the same effect as the above Delete command:

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

Purging does not affect any files in your UFD which only have one version. Note that the file specification for the Purge switch does not include a version field. After you issue the above command, you can issue the following command to see what files remain in your directory:

```
>PIP /LI<CR>

DIRECTORY DB0:[301,3143]
14-FEB-79  09:10

ADD.OBJ;2              2.                    25-JAN-79 14:31
ADD.MAP;1              4.                    25-JAN-79 14:32
ADD.STB;1              3.                    25-JAN-79 15:26
ADD.FTN;1              1.                    20-JAN-79 12:08
ADD.TSK;3             29.       C           25-JAN-79 14:33

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

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

4.2.6 Selectively Deleting Files

The PIP selective delete command prompts for your response before deleting a file. The responses allowed (<CR>, ^Z,Y,N,Q, and G) cause the following action:

Letter	Terminator	Operation
Y	<CR>	Deletes file and continues
Y	^Z	Deletes file and exits from PIP
N	<CR>	Saves file and continues
N	^Z	Saves file and exits from PIP
NULL (no response)	<CR>	Saves file and continues
NULL	^Z	Saves file and exits
Q	<CR>	Saves file and returns to command mode.
Q	^Z	Saves file and exits from PIP
G	<CR>	Deletes this and all remaining candidates for deletion, lists deleted files, and returns to PIP command mode.
G	^Z	Deletes this and all remaining candidates for deletion, lists deleted files, and returns to MCR.

You can specify the Selective Delete switch (/SD) as follows:

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

PIP responds:

```
DELETE FILE DB1:[303,12]ADD.FTN;1 [Y/N/G/Q]? Y
DELETE FILE DB1:[303,12]ADD.OBJ;1 [Y/N/G/Q]? N
.
.
.
```

In response to your answers, PIP deletes ADD.FTN;1 but does not delete ADD.OBJ;1.

4.2.7 Copying Files

Copying files is PIP's default function; that is, if you enter a legal PIP command line with no switches, PIP performs a copy operation. For example, the following command copies the file ADD.MAP from your UFD on SY: to your UFD on DK:.

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

The command includes the call to PIP, followed by a file specification in the form:

outfile=infile

infile

The file to be copied

outfile

The new copy of the file

When you omit the UFD, file name, file type, and/or version number in outfile, PIP defaults the UFD to your default UFD and the name, type, and version number of the file to the equivalent fields in the input file.

Before you can copy a file to a directory on another volume, the directory must exist on that volume. 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 volume specified in the above example does not contain a UFD corresponding to your UIC, PIP returns the message:

```
PIP -- CANNOT FIND DIRECTORY FILE
       [n,m]
```

4.2.8 Renaming Files

The PIP Rename (RE) switch allows you to rename existing files. For example:

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

This command tells PIP to change the names 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 input and output version fields

when you use the Rename switch. The wildcards in the output file specification indicate that the type and version of the renamed files remain the same. The directory now contains the following entries:

```
DIRECTORY DB0:[301,314]
4-AUG-78  3:03

ADDTWO.OBJ;2              2.                  25-JAN-79 14:31
ADDTWO.MAP;1              4.                  25-JAN-79 14:32
ADDTWO.STB;1              3.                  25-JAN-79 14:34
ADDTWO.FTN;1              1.                  20-JAN-79 15:26
ADDTWO.TSK;3             29.       C          19-JAN-79 02:45

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 will be named ADDTWO.

4.3 THE QUEUE MANAGER

System users often need to have copies of their files printed on the line printer. Consequently, the line printer is in heavy demand. An RSX-11M task called the Queue Manager provides for orderly use of the line printer. It does this by maintaining a queue of files to be printed according to their priority, time of request, and other factors, and by displaying information to let you know your file's place in the queue.

4.3.1 Printing Files

To send files to the line printer queue, you use the Queue Manager PRINT command, as follows:

```
>PRINT ADD.MAP<CR>
```

Note that you do not need to specify LP: or any other output file. This command asks the Queue Manager to enter ADD.MAP to the queue of files waiting to be printed. The system maintains the queue by file name and file identification number, copying each file or group of files to the line printer, usually on a first queued, first printed basis. The system also allows you to specify that files be printed after a certain time or date. Check the *RSX-11 Utilities Manual* for more information about the Queue Manager.

4.3.2 Listing Files in the Queue

To determine whether a particular file is in the queue, waiting to be printed, type the following Queue Manager command:

```
QUE /LIST<CR>
```

The names of all the files waiting to be printed will be displayed on your terminal in response to this command in the following format:

```
** PRINT QUEUES **
❶ PRINT => LP0:
[303,30]    MPREF     (2000,3276)        ACTIVE ON LP0:
   ❷ DB0:[303,4]MREF.DOC;1
     DB0:[302,310]ADD.FTN;1
     DK1:[100,1]ADD3.FTN;2
```

❶ Identifies the file currently being printed, its file identification number, name, and the printer.

❷ Names of the files waiting in the queue to be printed. (These are listed in the order in which they will be printed.)

APPENDIX A
SAMPLE FORTRAN PROGRAM

The following listing shows the development of the FORTRAN-IV program, ADD.FTN, and the manipulation of the resulting files. The listing illustrates how rapidly you can create numerous versions of the same file, and what to do to eliminate the old versions.

What you type on your terminal is printed in red; what the computer prints on your terminal is printed in black.

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

          RSX-11M BL26 MULTI-USER SYSTEM

GOOD MORNING
12-JAN-79  19:37  LOGGED ON TERMINAL TT32:

12-JAN-79

              SYSTEM WILL BE DOWN TODAY FROM 13:00-15:00

>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^R<CR>
          ACCEPT 2,K,L<CR>
2         FORMAT (222\22\I5)<CR>
          PRINT ^U
          TYPE 3,K+L<CR>
3         FORMAT (' THE SUM IS ',I5)<CR>
          STOP<CR>
          END<CR>
<CR>
*EXIT
[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>
[00008 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>
[00008 LINES READ IN]
[PAGE     1]
*INSERT<CR>
C        THIS PROGRAM ADDS TWO NUMBERS<CR>
<CR>
*<ESC>
<*BOB*>
*<CR>
C        THIS PROGRAM ADDS TWO NUMBERS
*LOCATE NUMBERS<CR>
1        FORMAT (' TYPE TWO NUMBERS - M,N')
*ADD !PROMPT FOR INPUT<CR>
*PRINT<CR>
1        FORMAT (' TYPE TWO NUMBERS - M,N')  !PROMPT FOR INPUT
*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
*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]


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


DIRECTORY DB1:[303,12]
6-MAY-79 20:09

ADD.FTN;3                1.          06-MAY-79  20:08
ADD.FTN;2                1.          06-MAY-79  20:07
ADD.FTN;1                1.          06-MAY-79  19:47

TOTAL OF 3./9. BLOCKS IN 3. FILES
```

```
>FOR ADD=ADD<CR>
.MAIN.
>PIP ADD.*;* /LI<CR>


DIRECTORY DB1:[303,12]
6-MAY-79 20:10

ADD.FTN;1                1.          06-MAY-79  19:47
ADD.FTN;2                1.          06-MAY-79  20:07
ADD.FTN;3                1.          06-MAY-79  20:08
ADD.OBJ;1                2.          06-MAY-79  20:09

TOTAL OF 5./20. BLOCKS IN 4. FILES


>FOR ,ADD/-SP=ADD<CR>
.MAIN.
>PIP ADD.*;* /LI<CR>


DIRECTORY DB1:[303,12]
6-MAY-79  20:10

ADD.FTN;1                1.          06-MAY-79  19:47
ADD.FTN;2                1.          06-MAY-79  20:07
ADD.FTN;3                1.          06-MAY-79  20:08
ADD.OBJ;1                2.          06-MAY-79  20:09
ADD.LST;1                2.          06-MAY-79  20:10

TOTAL OF 7./25. BLOCKS IN 5. FILES


>FOR ,ADD=ADD<CR>
.MAIN.
>PIP ADD.*;* /LI<CR>


DIRECTORY DB1:[303,12]
6-MAY-79  20:10

ADD.FTN;1                1.          06-MAY-79  19:47
ADD.FTN;2                1.          06-MAY-79  20:07
ADD.FTN;3                1.          06-MAY-79  20:08
ADD.OBJ;1                2.          06-MAY-79  20:09
ADD.LST;1                2.          06-MAY-79  20:10
ADD.LST;2                2.          06-MAY-79  20:12

TOTAL OF 9./30. BLOCKS IN 6. FILES
```

```
>FOR ADD,ADD/-SP=ADD<CR>
.MAIN.
>PIP ADD.*;* /LI<CR>


DIRECTORY DB1:[303,12]
6-MAY-79  20:12

ADD.FTN;1                1.          06-MAY-79  19:47
ADD.FTN;2                1.          06-MAY-79  20:07
ADD.FTN;3                1.          06-MAY-79  20:08
ADD.OBJ;1                2.          06-MAY-79  20:09
ADD.LST;1                2.          06-MAY-79  20:10
ADD.LST;2                2.          06-MAY-79  20:12
ADD.OBJ;2                2.          06-MAY-79  20:12
ADD.LST;3                2.          06-MAY-79  20:12

TOTAL OF 13./40. BLOCKS IN 8. FILES


.FOR ADD,ADD=ADD<CR>
.MAIN.
>PIP ADD.*;* /LI<CR>


DIRECTORY DB1:[303,12]
6-MAY-79  20:15

ADD.FTN;1                1.          06-MAY-79  19:47
ADD.FTN;2                1.          06-MAY-79  20:07
ADD.FTN;3                1.          06-MAY-79  20:08
ADD.OBJ;1                2.          06-MAY-79  20:09
ADD.LST;1                2.          06-MAY-79  20:10
ADD.LST;2                2.          06-MAY-79  20:12
ADD.OBJ;2                2.          06-MAY-79  20:12
ADD.LST;3                2.          06-MAY-79  20:12
ADD.OBJ;3                2.          06-MAY-79  20:13
ADD.LST;4                2.          06-MAY-79  20:13

TOTAL OF 17./50. BLOCKS IN 10. FILES
```

```
>TKB ADD=ADD<CR>
TKB -- *DIAG*-13 UNDEFINED SYMBOLS SEGMENT ADD

        ADI$MM
        EOL$
        ICO$
        IFR$
        IFW$
        ISN$
        LSN$
        MOI$MM
        REL$
        STP$
        TVI$
        $OTI
        $OTSVA



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


DIRECTORY DB1:[303,12]
6-MAY-79  20:16

ADD.FTN;1                1.              06-MAY-79  19:47
ADD.FTN;2                1.              06-MAY-79  20:07
ADD.FTN;3                1.              06-MAY-79  20:03
ADD.OBJ;1                2.              06-MAY-79  20:09
ADD.LST;1                2.              06-MAY-79  20:10
ADD.LST;2                2.              06-MAY-79  20:12
ADD.OBJ;2                2.              06-MAY-79  20:12
ADD.LST;3                2.              06-MAY-79  20:12
ADD.OBJ;3                2.              06-MAY-79  20:13
ADD.LST;4                2.              06-MAY-79  20:13
ADD.TSK;1                4.      C       06-MAY-69  20:14

TOTAL OF 21./54. BLOCKS IN 11. FILES
```

```
>TKB ,ADD/-SP=ADD,LB:[1,1]FOROTS/LB<CR>
>PIP ADD.*;* /LI<CR>

DIRECTORY DB1:[303,12]
6-MAY-79  20:18

ADD.FTN;1                1.                  06-MAY-79  19:47
ADD.FTN;2                1.                  06-MAY-79  20:07
ADD.FTN;3                1.                  06-MAY-79  20:08
ADD.OBJ;1                2.                  06-MAY-79  20:09
ADD.LST;1                2.                  06-MAY-79  20:10
ADD.LST;2                2.                  06-MAY-79  20:12
ADD.OBJ;2                2.                  06-MAY-79  20:12
ADD.LST;3                2.                  06-MAY-79  20:12
ADD.OBJ;3                2.                  06-MAY-79  20:13
ADD.LST;4                2.                  06-MAY-79  20:13
ADD.TSK;1                4.        C         06-MAY-69  20:14
ADD.MAP;1               14.                  06-MAY-79  20:14

TOTAL OF 35./69. BLOCKS IN 12. FILES

>TKB ,ADD,ADD=ADD,LB:[1,1]FOROTS/LB<CR>
>PIP ADD.*;* /LI<CR>

DIRECTORY DB1:[3003,12]
6-MAY-79  20:20

ADD.FTN;1                1.                  06-MAY-79  19:47
ADD.FTN;2                1.                  06-MAY-79  20:07
ADD.FTN;3                1.                  06-MAY-79  20:08
ADD.OBJ;1                2.                  06-MAY-79  20:09
ADD.LST;1                2.                  06-MAY-79  20:10
ADD.LST;2                2.                  06-MAY-79  20:12
ADD.OBJ;2                2.                  06-MAY-79  20:12
ADD.LST;3                2.                  06-MAY-79  20:12
ADD.OBJ;3                2.                  06-MAY-79  20:13
ADD.LST;4                2.                  06-MAY-79  20:13
ADD.TSK;1                4.        C         06-MAY-69  20:14
ADD.MAP;1               14.                  06-MAY-79  20:14
ADD.MAP;2               14.                  06-MAY-79  20:15
ADD.STB;1                4.                  06-MAY-79  20:12

TOTAL OF 53./89. BLOCKS IN 14. FILES
```

```
>TKB ADD,ADD,ADD=ADD,LB:[1,1]FOROTS/LB<CR>
>PIP ADD.*;* /LI<CR>


DIRECTORY DB1:[303,12]
6-MAY-79  20:13

ADD.FTN;1                1.              06-MAY-79  19:47
ADD.FTN;2                1.              06-MAY-79  20:07
ADD.FTN;3                1.              06-MAY-79  20:08
ADD.OBJ;1                2.              06-MAY-79  20:09
ADD.LST;1                2.              06-MAY-79  20:10
ADD.LST;2                2.              06-MAY-79  20:12
ADD.OBJ;2                2.              06-MAY-79  20:12
ADD.LST;3                2.              06-MAY-79  20:12
ADD.OBJ;3                2.              06-MAY-79  20:13
ADD.LST;4                2.              06-MAY-79  20:13
ADD.TSK;1                4.       C      06-MAY-69  20:14
ADD.MAP;1               14.              06-MAY-79  20:14
ADD.MAP;2               14.              06-MAY-79  20:15
ADD.STB;1                4.              06-MAY-79  20:12
ADD.TSK;2               33.       C      06-MAY-79  20:15
ADD.MAP;3               14.              06-MAY-79  20:15
ADD.STB;2                4.              06-MAY-79  20:13

TOTAL OF 104./142. BLOCKS IN 17. FILES


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


DIRECTORY DB1:[303,12]
6-MAY-79  20:17

ADD.FTN;3                1.              06-MAY-79  20:08
ADD.OBJ;3                2.              06-MAY-79  20:13
ADD.LST;4                2.              06-MAY-79  20:13
ADD.TSK;2               33.       C      06-MAY-79  20:15
ADD.MAP;3               14.              06-MAY-79  20:15
ADD.STB;2                4.              06-MAY-79  20:13

TOTAL OF 56./68. BLOCKS IN 6. FILES
```

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


DIRECTORY DB1:[303,12]
6-MAY-79 20:21

ADD.FTN;3                1.                  06-MAY-79  20:08
ADD.OBJ;3                2.                  06-MAY-79  20:13
ADD.LST;4                2.                  06-MAY-79  20:13
ADD.TSK;2                33.       C         06-MAY-79  20:15
ADD.MAP;3                14.                 06-MAY-79  20:15
ADD.STB;2                4.                  06-MAY-79  20:13

TOTAL OF 56./56. BLOCKS IN 6. FILES
```

Note that the last step reduced the number of blocks allocated to the 6 files in the directory.

```
>RUN ADD<CR>
TYPE TWO NUMBERS - M,N
8,9<CR>
THE SUM IS       17
TT32: -- STOP


>RUN ADD<CR>
TYPE TWO NUMBERS - M,N
43,54<CR>
THE SUM IS       97
TT32: -- STOP


>RUN ADD<CR>
TYPE TWO NUMBERS - M,N
998,2<CR>
THE SUM IS 1001
TT32: -- STOP
```

A

B

C

E

F

I

L

M

N

O

P

Q

R

S

T

U

V

W

RSX-11M Beginner's Guide
AA-5245B-TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

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
- ☐ Other (please specify)

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

RT/C SOFTWARE PUBLICATIONS TW/A14
DIGITAL EQUIPMENT CORPORATION
1925 ANDOVER STREET
TEWKSBURY, MASSACHUSETTS 01876

Do Not Tear - Fold Here

Cut Along Dotted Line

RSX–11M

Mini–Index

Order No. AA-H262A-TC

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, May 1979

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

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.

The following are trademarks of Digital Equipment Corporation:

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-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

INTRODUCTION

This manual is a brief index to the RSX-11M documentation set. It indicates where in the set information on a general topic can be found. A more detailed master index is being published as a post-release document. Neither index, however, is intended to replace the indexes found in the individual RSX-11M manuals.

An acronym printed in italic capital letters follows each entry in this index, and represents the name of the manual in which the information can be found. Chapter and section number references follow the acronym. Some entries are followed only by the chapter number(s), for example:

PIP (Peripheral Interchange Program), *UTL* 4

This entry indicates that PIP is discussed throughout chapter 4 of the utilities manual.

Following is a list of the acronyms with the corresponding manual titles. (The RSX-11M/RSX-11S Documentation Directory lists the order numbers for these manuals.)

BEG — RSX-11M Beginner's Guide

CDA — RSX-11M/M-PLUS Crash Dump Analyzer Reference Manual

DEV — RSX-11M/M-PLUS Guide to Program Development

DRV — RSX-11M/M-PLUS I/O Drivers Reference Manual

ERL — RSX-11M/M-PLUS Error Logging Reference Manual

EXE — RSX-11M/M-PLUS Executive Reference Manual

GEN — RSX-11M System Generation and Management Guide

INT — Introduction to RSX-11M

IOP — IAS/RSX-11 I/O Operations Reference Manual

MAC — IAS/RSX-11 MACRO-11 Reference Manual

MCR — RSX-11M/M-PLUS MCR Operations Reference Manual

ODT — IAS/RSX-11 ODT Reference Manual

SLR — IAS/RSX-11 System Library Routines Reference Manual

TKB — RSX-11M/M-PLUS Task Builder Manual

UMD — RSX-11M/M-PLUS User Mode Diagnostics Reference Manual

UTL — RSX-11 Utilities Manual

WRT — RSX-11M Guide to Writing an I/O Driver

INDEX

A

B

C

INDEX

D

INDEX

INDEX

INDEX

F

INDEX

G

H

I

K

L

M

INDEX

N

O

P

INDEX

INDEX

T

INDEX

U

INDEX

RSX-11M
Mini-Index
AA-H262A-TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

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
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code or Country_____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

RT/C SOFTWARE PUBLICATIONS TW/A14
DIGITAL EQUIPMENT CORPORATION
1925 ANDOVER STREET
TEWKSBURY, MASSACHUSETTS 01876

Do Not Tear - Fold Here

Cut Along Dotted Line