pdp11

RSTS-11
System User's Guide

Order No. DEC-11-ORSUA-D-D

digital

RSTS-11

System User's Guide

Order No. DEC-11-ORSUA-D-D

digital equipment corporation · maynard. massachusetts

First Printing, September 1972
First Revision, May 1973
Second Revision, September 1974
Third Revision, July 1975

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 to the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

Copyright © 1972, 1973, 1974, 1975 by Digital Equipment Corporation

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		

LIMITED RIGHTS LEGEND

Contract No. _____

Contractor or Subcontractor: Digital Equipment Corporation

All the material contained herein is considered limited rights data under such contract.

CONTENTS

FIGURES

TABLES

TABLES (Cont.)

PREFACE

This manual contains information concerning operating characteristics of RSTS-11 while running under Version 4 (RSTS), or Versions 5 and 6 (RSTS/E). The manual deals with the interaction between the user and the RSTS-11 terminal, describes how to enter and edit BASIC-PLUS programs, explains system commands and system programs, and tells how to operate certain system hardware. For more information on RSTS/E documentation, consult the RSTS/E Documentation Directory.

This manual describes the following versions of RSTS-11 operating systems released on the dates shown.

RSTS V4A-12	October 1972
RSTS/E VØ5-21	July 1973
RSTS/E VØ5B-24	September 1974
RSTS/E VØ5C-Ø1	February 1975
RSTS VØ4B-17	July 1975
RSTS/E VØ6A-Ø2	July 1975

Features not included on certain versions are indicated by footnotes in the text. To determine the version of a given system, simply type the HELLO command after which the LOGIN program runs and prints the system identification line. The first item on the line is one of the version numbers described above. On systems which print other than RSTS version numbers, the user must consult the system manager for the correct version number.

CHAPTER 1
INTRODUCTION TO RSTS-11

RSTS-11 (Resource Sharing Timesharing System for the PDP-11/20, PDP-11/40, PDP-11/45 and PDP-11/70 computers) is a powerful timesharing system that, with sufficient hardware, can support up to 64 interactive terminals. From each terminal a user has access to any peripheral device on the system.

RSTS-11 uses the BASIC-PLUS language, an enriched version of the BASIC language as originally developed at Dartmouth College. BASIC-PLUS is compatible with existing BASIC programs and includes facilities to handle strings, matrices, and files. Also, COBOL is optionally available under RSTS/E.

1.1 TIMESHARING

Early computers were the province of the mathematician. Used mainly to solve differential equations, the systems were narrow in scope and poorly utilized. Since few persons were knowledgeable enough to employ the enormous processors, one individual could monopolize computer time -- sit at the console and solve problems in step-by-step fashion.

As more people discovered computing techniques, it was no longer practical to let a few persons monopolize computer time. To increase machine efficiency, batch processing was introduced. In this mode of operation, no time was wasted between jobs. Programs were punched on cards and the cards stacked and fed to the computer in batches. Operation of each program was governed by control cards that took the place of the human operator.

Since card reading is a relatively slow process, some early systems employed a small computer to read the cards and transfer program information to magnetic tape that was then input to the large computer. As a further refinement, programs were assigned priorities, with short jobs being executed first to minimize job turnaround.

But what about the computer user? As computer utilization improved, program development took more time. To develop a new program, a user performed the following procedure. After writing the program on paper, he carried it to a keypunch operator to have the cards punched and verified. A day or so later, when the program was returned, the user checked for punching errors, then returned to the keypunch for corrections.

Next, he sent the cards to the computer center for compilation. The compilation, which might not be returned for a half day or more, could reveal spelling or syntactical errors. The cards then had to be changed and resubmitted -- another half day's wait. If the next

compilation was successful and the program was run, program logic errors might be discovered -- new cards, new compilation, etc., etc. In addition, the user often studied reams of computer listings to find the errors. Using these inefficient methods, even simple programs might take weeks to develop.

Batch processing maximizes machine efficiency in routine data processing operations where turnaround is not critical. But for program development and modification, the user requires another mode of operation. The user needs a way to "interact" with the computer -- to feed his program to the system, line by line, and continuously check the results.

In fact, the user may want to develop interactive programs. These programs, which are extremely productive tools, ask the user questions and perform an analysis based on his answers.

If the user had unlimited funds, he might be tempted to buy or lease a large computer -- a system he could dedicate to his work that would provide sufficient power, many peripherals, and a large variety of software. With such a system, he could develop programs interactively or utilize batch processing for routine tasks. However, costs normally preclude the dedication of a large system to a single user.

By using timesharing, the user has most of the benefits of a dedicated system at a small fraction of the cost. Timesharing with today's technology allows a large powerful computer such as a PDP-11/70 to handle 16, 32, or more users simultaneously. Through a choice of terminals, the user can interact with the system or initiate batch processing which runs concurrently. The user also has access to a choice of mass storage and peripherals and a selection of application programs. Since response is fast, the user appears to have a dedicated system. Yet costs are shared. He pays only for the time and facilities that he requires and doesn't pay for the time the machine is idle.

A timesharing system isn't just any computer with some additional hardware and software. It's a system designed specifically for timesharing. Otherwise, facilities are limited, fewer users can be handled efficiently, and economics are unattractive.

In a simple timesharing system, each program is assigned a fixed time slice or time quantum and operation is switched from one program to another in round robin fashion until each program is completed. Essentially, if each user receives 1/60 of a second and 12 users are "on" the system, each user will receive service every 1/5 of a second.

The timesharing system performs multiprogramming; that is, it allows several programs to reside in core simultaneously and to operate sequentially. The switching between programs is initiated by a clock which interrupts the central processor to signal that a certain time period

has elapsed. A monitor, also called an operating system or executive program, directs the execution of these tasks and performs other duties.

The system discussed so far services a number of users sequentially in round robin fashion. To increase the number of users serviced, more main memory or core is required. However, since core is expensive, a secondary memory is employed. This memory -- usually magnetic disk or drum -- is slower than core or main memory but provides greatly increased capacity at reasonable cost. User programs can be located in secondary memory and moved into main memory for execution. Programs entering main memory exchange places with a program (or programs) that has just been serviced by the central processor. This operation is called swapping.

In operation, main memory is divided into separate memory blocks. Secondary memory is connected to these blocks through a high speed input/output processor -- a hardware device that allows the disk or drum to swap a program into any one of the main memory blocks without any aid from the central processor. This structure allows the central processor to be operating a user program in one block of memory while programs are being swapped to and from another block. This independent overlapped operation greatly improves efficiency and processing power.

Round robin scheduling, in which each program operates in sequence and receives a fixed amount of time, is effective only if all programs have similar requirements. Such is not the case, however. At any particular time, a timesharing system will be handling some programs which require extensive amounts of computing time (and are said to be compute bound) and other programs that must stop frequently for input or output (I/O bound).

To serve programs at and between these two extremes, the scheduling algorithm must provide frequent service to I/O bound programs and must give compute bound jobs longer time quantums to prevent wasteful swapping. A simple dynamic scheme could provide two queues -- one for each type of job. When a user first logs on to the system, he is placed in an I/O bound queue (waiting line) where he receives frequent service and small time quantums. If the program isn't completed or does not request input or output during the time allotted to him, the job needs more computing time and is placed in the compute bound queue. Thus the scheduling algorithm optimizes system efficiency by automatically adjusting to program requirements.

In the present state of scheduling art, algorithms are constantly being changed and improved. Current algorithms are extremely sophisticated, providing excellent service for most timesharing job mixes. They also allow fine tuning, if such modifications are necessary.

The ability of the algorithm to match processing to program requirements ensures the best service possible for all user programs.

A timesharing system has performed its basic function if it allows a number of users simultaneous access to a central computer. However, to be fully useful, the system should also allow the users access to other system resources -- storage devices for his programs and data, line printers, card readers, etc. For example, the user should be able to choose between magnetic tape and disk for program storage. And if he has a 50-page report to produce, he should be able to employ a line printer instead of his terminal. If users controlled these devices, however, much confusion might result. For example, two users might select the line printer at the same time. If one user was processing Abraham Lincoln's Gettysburgh Address and another Mark Anthony's funeral oration, the report might look like the following:

```
I COME TO BURY CAESAR NOT TO PRAISE HIM
FOUR SCORE AND SEVEN YEARS AGO
THE EVIL THAT MEN DO LIVE AFTER THEM
OUR FATHERS BROUGHT FORTH ON THIS CONTINENT
```

To prevent users from interfering with each other, the monitor coordinates input and output (I/O).

The user can request that a particular device be assigned for his use, and release the device upon completion of an operation. During this time no other user can perform I/O to an assigned device. If the user does not specifically assign a device but attempts to use it, he is given access to the device if it is not already assigned to another user or currently being used.

If a user does not require a fast device for his exclusive use (like a private disk), he can elect to use a public device, such as the public disk area, line printer, paper tape devices, etc.

User programs and data can coexist on the same storage device. A filing system is necessary if program and data segments are to be retrieved in proper order.

Data is transferred from memory to a peripheral device as a block of words or a record. (A word is the number of binary digits or bits that the central processor can retrieve and "operate on" at one time.) Record length can be arbitrary or dictated by the physical device being used, for example, the number of columns on an 80 column card or on a 132 column line printer. For PDP-11 disk files, the normal length is 256 words. This can be altered with the RSTS-11 Record I/O features.

For convenience each user's blocks are organized in groups called files. Files, like memory, must be protected from access by unauthorized users. When a user creates a file, he can restrict it, specifying whether others can have access, and (if access is permitted) whether the files can be modified or only read. With such an arrangement, programmers in various locations can use the same data to work simultaneously on the same project; but unauthorized personnel cannot modify or read the files.

Users can communicate with the computer over the conventional dial-up telephone network. User terminals can, therefore, be located anywhere that phone service is available and be connected to any computer system, feasibility limited only by long distance phone rates.

Each user terminal is connected to a data set or modem (modulator-demodulator) which converts user terminal output into a signal suitable for the telephone network. At the computer end of the phone lines, there is another data set which reconverts the signal and feeds it to the central processor.

The number of data sets employed at the user end of the system is unlimited. At the computer end of the communications network, however, the number of data sets is limited by the number of users that can be serviced simultaneously by the system.

In order to gain access to the system, the user dials the system phone number from his data set. The telephone network handles the call, scanning the data sets at the computer system. If all of the sets are busy, the user receives a busy signal, just as he would with normal phone service. If a set is available, the telephone network rings it, and the computer answers the call, placing the user in communication with the monitor. The terminal is then on-line and ready for operation.

1.2 USING RSTS-11

The remainder of this document describes how to use the RSTS-11 system from the user terminal. The user may issue commands to the RSTS-11 monitor (Chapter 2) or use any of the system programs available in the system library (Chapter 4). Chapter 5 describes the manual operation of typical RSTS-11 peripheral devices.

For information on the BASIC-PLUS language, the user is referred to the BASIC-PLUS Language Manual. The appendices to this document summarize the language elements as well as other information such as commands and error messages.

CHAPTER 2
RSTS-11 SYSTEM COMMANDS

2.1 ON-LINE WITH RSTS-11

2.1.1 Project-Programmer Numbers and Passwords

Before the user attempts to use the RSTS-11 system, the system manager will assign him a unique project-programmer number and password. The project-programmer number might, for example, look as follows:

[1ØØ , 1Ø1]

The number 1ØØ above is the project number (possibly held by a group of people having a common interest) ; and the number 1Ø1 is the programmer number (held by only one person within the project group). Thus, each individual's project-programmer number is different. The project-programmer number is also called the user's account number. Protection codes are assigned to user files on the basis of the various relationships among users as determined by the components of their account numbers (see section 2.4.4 describing the NAME-AS statement).

The user password is a unique alphanumeric code assigned to an individual user. This password is never printed on the terminal and, hence, allows for a measure of security in limiting the use of the computer system.

2.1.2 Getting On-Line, HELLO Command

Equipped with the codes to obtain access to the system, the user should find a terminal and turn the LINE-OFF-LOCAL knob[1] to LINE. This puts the terminal on-line to RSTS-11, that is, opens a line of communication between the computer and the terminal.

Once the terminal is on-line, type the command:

HELLO

followed by the RETURN key. This tells RSTS-11 that a user wishes to join the system. RSTS-11 responds by printing a system identification message, then prints a number sign (#) at the left margin of the paper, and then waits for the user to type his project-programmer number (followed by the RETURN key). The system responds by printing:

PASSWORD:

and then waits for the user to type his password followed by the RETURN key. The password

---

[1] Alternate types of user terminals may have a different knob or switch designed to put the device on-line. See Chapter 5 for information concerning other devices.

characters are not printed at the console. If the codes are acceptable to the system, the user is logged into the system. A message specified by the system manager is then printed. This message generally changes from day to day and provides the user with information on any changes or additions to the system.

If the codes entered are incorrect, the message

INVALID ENTRY - TRY AGAIN

is printed and the user can try again.

The entire process of entering the system is shown below (although the RETURN key is typed to enter a line to the system, it does not echo on the terminal paper except to perform a carriage return/line feed operation). Characters printed by the system are underlined to differentiate them from characters typed by the user.

```
HELLO

RSTS V06A-01 SYSTEM #880  JOB 10  KB16  02-JUN-75  04:06 PM
#120,80
PASSWORD:

WE HAVE INSTALLED A NEW RP04 DISK ON DRIVE #1. WE ARE
ATTEMPTING TO RESTORE DRIVE #1 FILES AS OF MAY 15...


READY
```

Once the user is successfully logged onto the system, the READY message is printed. The system prints READY to indicate that the terminal is at BASIC-PLUS command level. The user can type NEW to create a new program, OLD to retrieve a program previously saved, or any other RSTS-11 command.[1]

An alternate way of logging into the system is to enter the project-programmer numbers on the same line with the HELLO command, as shown below. This results in the system prompting the typing of the password only.

[1]On Version 4 (RSTS) systems, the message NEW OR OLD is printed once the user is successfully logged onto the system. However, the user can type any RSTS-11 command.

```
HELLO 120,80
PASSWORD:

        RSTS TIMESHARING HOURS ARE:

            8:30 AM  TO  8:00 PM    MONDAY THRU FRIDAY


READY
```

Note that, on Version 4 (RSTS) systems, LOGIN prints the system identification message before printing PASSWORD.

Another option available when logging into the system is the use of the slash character rather than the comma to separate the project and programmer numbers. The slash character inhibits the opening message (s) printed by the system. Such messages are installed on the system by the system manager and generally contain useful information. However, when logging into the same system several times in one day, it is frequently desirable to eliminate the time necessary to print a message the user has seen previously.

```
HELLO 120/80
PASSWORD:


READY
```

2.1.3 Going Off-Line, BYE Command

When the user is ready to leave the terminal, he types the command:

BYE

followed by the RETURN key. This tells RSTS-11 that the user has requested to be dismissed from the system. At this point RSTS-11 prints

CONFIRM:

and waits for the user to give one of the following replies:

CONFIRM: Reply	Meaning
N	No. The logout sequence is terminated, the user is not logged out, and the system again replies READY.
I	Individual. The user wishes to individually examine each of his files. The system prints the name of each user file on the system disk, its size, protection code and creation date, followed by a ? character. The user can reply by typing K to kill (delete) the file or type the RETURN key to retain the file. (Any reply other than K causes the file to be retained.)
?	Causes the system to print an explanation of the possible CONFIRM: replies.
Y	Yes. The system attempts to proceed with the logout sequence, checking to see that the user has not exceeded the disk quota for his project-programmer number. If the user has used more than his share of disk storage, an appropriate message is printed and he is not allowed to leave the system until he is within his disk quota.
F	Fast logout. Same as Y except that the accounting messages are not printed.[1]

The following is the simplest case in which the user simply logs off the system.

```
BYE
CONFIRM: Y
SAVED ALL DISK FILES; 300 BLOCKS IN USE, 100 FREE
JOB 10 USER 120,80 LOGGED OFF KB15 AT 15-AUG-74 03:26 PM
SYSTEM RSTS V05B-24 SYSTEM #880
RUN TIME WAS 5.3 SECONDS
ELAPSED TIME WAS 7 MINUTES, 58 SECONDS
GOOD AFTERNOON
```

The following example demonstrates the ? and N replies:

```
BYE
CONFIRM: ?
OPTIONS FOR 'CONFIRM:' ARE:
 ?       THIS HELP MESSAGE
 N       DON'T LOG ME OUT
 I       INDIVIDUAL FILE DELETION
          K TO DELETE
          <CR> TO SAVE
 F       FAST LOGOUT
CONFIRM: N


READY
```

[1] This feature is not available prior to Version 5 (RSTS/E) systems.

The following example shows the sequence in which the user is over his disk quota; deletes some files, and then logs out of the system:

```
BYE Y
DISK QUOTA OF 400 EXCEEDED BY 50 BLOCKS
SOME FILE(S) MUST BE DELETED BEFORE LOGGING OUT
TYPE '?' FOR HELP
CONFIRM: I
A       .BAS        150      60      15-AUG-74   ?
C       .BAS        150      60      15-AUG-74   ?
B       .BAS        150      60      15-AUG-74   ? K
TYPE '?' FOR HELP
CONFIRM: Y
SAVED ALL DISK FILES; 300 BLOCKS IN USE, 100 FREE
JOB 18 USER 120,80 LOGGED OFF KB15 AT 15-AUG-74 03:34 PM
SYSTEM RSTS V05B-24 SYSTEM #880
RUN TIME WAS 2.2 SECONDS
ELAPSED TIME WAS 3 MINUTES, 2 SECONDS
GOOD AFTERNOON
```

Upon logging the user out of the system, RSTS-11 deletes, from the disk, any temporary files which have been created by the system for the user. Any files created by the user and still remaining open on disk (or any I/O device) are closed and saved for future reference.

Before leaving the terminal, the user should turn the LINE-OFF-LOCAL knob[1] to OFF. (Turning the knob to LOCAL means that the terminal has power, but is not connected to the system. It then operates as a typewriter.)

2.1.4 Commands That Can Be Given Without Logging into the System

As long as the RSTS-11 system is in operation, anyone can turn a terminal on-line and give one of the commands described in Table 2-1.

Any of the commands in Table 2-1 can be issued from a terminal prior to logging into the system, although the full capabilities of the system program may only be available to a user already logged into the system. For details, see the appropriate section on the individual system program.

---

[1] Again, any other terminal being used should be turned off-line and powered down when not in use.

Table 2-1
Logged-Out RSTS-11 Commands

Command	Function
HELP	Causes a text file to be output to the user terminal explaining how to log into the system and various system features.
SYS	Causes the system to output a system status report to the user terminal, using the SYSTAT system program. See section 4.3.
SET xxxx	Allows the user to set the characteristics of the user terminal, using the TTYSET system program. xxxx is one of the acceptable TTYSET arguments, see section 4.5.
HELLO	Allows the user to enter the system. Requires the knowledge of a legal account number set and the associated password. Causes the LOGIN system program to be run, see section 4.1.
QUE /L dev:	Prints a listing of the jobs in the queue for the specified spooling programs. If device is not specified, the queue listing of LPØ: is printed.

2.2 CREATING A BASIC-PLUS PROGRAM

2.2.1 NEW Command

In order to create a new user program, the user issues the NEW command:

```
NEW
```

followed by the RETURN key. The system responds by printing:

```
NEW FILE NAME--
```

to which the user responds by typing the name of the new program. No filename extension is required (or accepted) at this point. The SAVE and COMPILE commands automatically append the correct filename extension.

Alternatively, the user can give the NEW command followed by the program name, to avoid having the system prompt typing of the program name. The command:

```
NEW PROG
```

is equivalent to the sequence:

```
NEW
NEW FILE NAME-- PROG
```

When the NEW command sequence is entered to the system, it:

a. Deletes any program currently in memory, and
b. Causes RSTS-11 to store the new program name.

The command of the form:

```
NEW DTØ:BOGLE
```

is meaningless. New programs are input from the user terminal only. The OLD command is used to input programs from other devices. No system check is made of an existing file of the name given in the NEW command. All checking for duplicate file names occurs when the SAVE command is given.

The user has the option of typing the RETURN key instead of indicating a filename. This causes RSTS-11 to create a file called NONAME which can be saved or compiled and referenced later as NONAME.BAS or NONAME. BAC. This name can be changed at any time (see sections 2.2.3 and 2.4.3). The creation of the file NONAME is shown below (the RETURN key, although typed, does not echo):

```
NEW
NEW FILE NAME--

READY
```

If the SAVE command is issued at this point, it will create the file NONAME.BAS.

2.2.2 Input of the New Program

Once the NEW command has been given and a new file created in memory, the user has the option of typing the program into the system or entering the program from a pre-punched paper tape through the terminal low-speed reader.[1]

If the user is typing his program into the system, he will likely want to use the RSTS-11 editing features (section 2.2.3) and the LIST and DELETE commands (sections 2.2.4 and 2.2.5). Information on how to use the low-speed terminal reader is found in sections 2.6.1 and 5.1.4.

[1]The ASR-33 Teletype terminal has a low-speed punched paper tape reader. RSTS-11 considers input from the Teletype paper tape reader equivalent to input from the terminal keyboard.

2.2.3 Editing BASIC-PLUS Programs

While typing a program at the terminal, or after a source program is brought into memory or run, changes can be made in the source program text. These changes are made in what is called the editing phase of BASIC, between the time when the system prints READY and the time when the user types RUN. (During this phase, system commands and immediate mode statements can be executed.)

The simplest type of correction is done during the typing of a line, before the line is entered to the system with the RETURN key. For example:

```
1Ø DEF FUN(X)
```

If the user realizes that he has typed FUN instead of FNU, he can type the RUBOUT key once for each character to be erased. The RUBOUT key causes the erased character to be echoed on the user terminal between backslashes as they are erased. For example:

```
1Ø DEF FUN(X<Rubout><Rubout><Rubout><Rubout>NU (X)=X↑2*X/2 + X/2
```

Typing the above is printed on the terminal as follows:

```
1Ø DEF FUN(X\X(NU\NU(X)=X↑2*X/2+X/2
```

If the RETURN key is typed at the end of the above line, the system would receive it as follows:

```
1Ø DEF FNU(X)=X↑2*X/2+X/2
```

Frequently it is easier to delete the entire line, rather than type the RUBOUT key several times. If the user has not yet entered the line to RSTS-11 with the RETURN key, he can type CTRL/U (see section 3.6), which erases the entire current physical line. If the RETURN key has been typed, the line can be retyped, and the second version will replace the first in the computer memory. For example:

```
2Ø LET X = 44.2 : A = 2Ø.Ø1 : B$ = "ABC"
2Ø LET X = 45.2 : A = 2Ø.Ø1 : B$ = "ABC"
```

If these two lines are typed in succession, only the second line is retained by the system.

2.2.4 LIST Command

The LIST command is used to obtain a clean printed copy of all or part of the user's current program at the user terminal. This is especially useful during and after an editing session in which the original program is changed.

In order to obtain a printed copy of the entire program as it currently exists within the system, type:

```
LIST
```

When listing the whole program, source lines are printed in line number sequence regardless of the order in which the lines were entered. In order to list a single line, type LIST followed by the line number.

```
LIST 1ØØ
```

obtains a printed copy of line 1ØØ.

In order to list a section of the program, type LIST followed by two line numbers separated by a dash.

```
LIST 1ØØ - 2ØØ
```

lists all program lines from line number 1ØØ to line number 2ØØ, inclusive. If 1ØØ and/or 2ØØ do not exist in the program, any lines within the range 1ØØ to 2ØØ are listed.

One or more single lines or program sections can be specified in a single LIST command by separating the individual elements with commas. For example:

```
LIST 25, 3Ø, 5Ø-75, 95, 1ØØ-15Ø
```

causes single lines 25, 3Ø, and 95 to be printed along with the program lines between 5Ø and 75 and between 1ØØ and 15Ø . Lines or program sections need not be indicated in sequential order in the LIST command, but the printed lines appear in the order requested.

In each of the previous examples, BASIC prints a program header containing the program name and the current system date and time. If this header material is not desired (as it might not be

during normal editing), the command may be given as LISTNH to delete the header material. To summarize:

LIST Command	Meaning
LIST	List the entire user program as it currently exists.
LISTNH	Same as LIST, but without the program header.
LIST n	List line n.
LISTNH n	List line n without the program header.
LIST n1-n2	List lines n1 through n2, inclusive
LISTNH n1-n2	List lines n1 through n2, inclusive, without the program header.

Extensive examples of program listings are shown in the BASIC-PLUS Language Manual.

In listing a program, the ? character is printed at the left of each line which RSTS-11 considers to be in error. For example:

```
LISTNH
10 LET A,B = 25
?20 PPRINT A+B
    .
    .
    .
```

The LIST command sends output to the user terminal only. If a line printer is available on the system,

```
SAVE LP:
```

is the fastest way to obtain a complete program listing. (See Section 2.4.1.)

2.2.5 DELETE Command

The DELETE command is used to remove one or more lines from the current user program. For example:

```
DELETE 100
```

causes line number 100 to be deleted.

```
DELETE 100 - 200
```

causes all program lines between and including line numbers 100 and 200 to be deleted. If 100 and/or 200 do not exist in the program, any lines within the range from 100 to 200 are deleted.

If several lines or groups of lines are to be deleted, the user can separate the individual elements with commas, as follows:

```
DELETE 1ØØ-2ØØ, 255, 3ØØ-4ØØ, 47Ø, 1ØØØ-11ØØ, 475
```

which deletes all lines between 1ØØ and 2ØØ, line 255, lines 3ØØ to 4ØØ, lines 47Ø and 475, and lines 1ØØØ to 11ØØ. Lines or program segments to be deleted need not be listed in sequential order in the DELETE command.

If only one line is to be deleted, it may be more convenient merely to type the line number and the RETURN key as follows:

```
1Ø
```

which is equivalent to:

```
DELETE 1Ø
```

Before deleting any line, the user should be certain that no other line references the deleted line (such as a GOTO statement), unless the deleted line is to be replaced. A reference to a missing line number will generate an error message when the program is run, halting execution.

If all lines from the current program are to be deleted, type the DELETE command without a number. Remember: typing the DELETE command without specifying a line number deletes all lines of the current program

2.2.6 RENAME Command

The RENAME command causes the name of the program currently in core to be changed to the specified name. For example:

```
RENAME  NEWNAM
```

The old name of the program currently in memory is discarded. The current program is now known as NEWNAM. If the SAVE command is given at this point:

```
SAVE
```

the file NEWNAM.BAS is stored on the system disk.

2.2.7 Debugging BASIC-PLUS Programs

The phase of program development during which the user is testing the program is called the debugging phase . Rather than repeatedly executing the program with minor alterations on each cycle, debugging can be facilitated by placing STOP statements at strategic places throughout the program. When the program is executed, each STOP statement causes a program halt, and the message:

```
STOP AT LINE 5Ø
```

is printed. In the above case, the STOP statement was located at line 5Ø. The user at a RSTS-11 terminal can then use immediate mode instructions to examine and/or change data values.

Issuing the CONT command causes program execution to continue at the next statement following the last STOP statement executed.

Once a program is successfully debugged, the extraneous STOP statements can be removed with the DELETE command.

Typing CTRL/C (see Section 3.5) also causes program execution to halt, but there is less control over where the program halts. The system prints READY and the CONT command can be issued. In this case the program may or may not be able to continue depending upon the program status when the CTRL/C was entered. On some systems, the variable LINE contains the line number of the statement being executed when the program halts. See Chapter 4 of the BASIC-PLUS Language Manual for more information on program debugging and interruption.

When debugging a program with considerable amounts of terminal output, CTRL/O can be used as a switch to stop and restart such output (see Section 3.7).

2.2.8 CONT Command

As explained in Section 2.2.7, the STOP statement and CTRL/C can be used to cause execution halts in a user program. Immediate mode examination of values or changes can then be performed, and the program restarted at the statement following the last executed statement by giving the

```
CONT
```

command, followed by the RETURN key. When the CONT command is entered, the system attempts to continue program execution whenever possible. (This is also true of program halts due to execution errors. Following the printing of some error messages, the CONT command may cause program execution to be resumed.) If it is not possible to continue program execution for

any reason the

CAN'T CONTINUE

message is printed. In almost all cases, it is possible to continue program execution once a CTRL/C or STOP command is executed.

2.2.9 CCONT Command

The CCONT command performs the same actions as the CONT command but additionally detaches the job. This command is useful for continuing a lengthy job that does not involve further terminal interaction. The CCONT command is available only to privileged users. An attempt by a non-privileged user to use the CCONT command results in the ILLEGAL SYS( ) USAGE message. To continue, the non-privileged user must use the CONT command.

2.3 EXECUTING A BASIC-PLUS PROGRAM

2.3.1 RUN Command

The RUN command is used to cause the execution of any source or compiled BASIC-PLUS program under RSTS-11. (Source programs are stored as typed by the user; compiled programs are described in Section 2.3.3.)

In order to run the program currently in memory, the user simply types:

RUN

This causes the execution of the program. A program header is printed after the RUN command is given, consisting of the program name and the current system date and time. If this information is not desired, the command:

RUNNH

should be given. RUNNH executes the current program without printing the header material. Any characters or filename specified in the RUNNH command are ignored; only the program currently in memory is run.

Where it is desired to run a program not currently in memory, the command:

RUN FILEV3

can be given. This command causes RSTS-11 to search for the file FILEV3.BAC or FILEV3.BAS on the system disk; load, compile (if necessary), and run it (if the file is found). (Since blanks are not significant in BASIC-PLUS commands, the RUN filename command does not execute properly if the first two characters of the filename are NH. For example, RUN NHTAX results in the execution of a RUNNH command, and the program currently in memory is run.)

If FILEV3.BAS (source) and FILEV3.BAC (compiled) both exist, RSTS-11 loads and executes FILEV3.BAC since it requires less time. In order to retrieve and execute FILEV3.BAS, it is necessary to issue separate OLD and RUN commands. The file is then available for any editing to be performed.

If FILEV3.BAS exists but FILEV3.BAC does not, then the command

```
RUN FILEV3
```

loads, compiles and executes FILEV3.BAS.

Where the file to be run is not present on the system disk, but is stored on another device, the format:

```
RUN dev:FILNAM
```

is used where dev: is the designation of the storage device. For example:

```
RUN DT1: TRANS
```

which runs the file TRANS.BAS found on DECtape unit 1. The statement:

```
RUN PR:
```

reads a BASIC program from the high speed reader and runs it. Since PR: is an input-only (non-file structured) device, no filename need be specified.

If a filename is specified with a non-file structured device in the RUN command, that filename is used as the current program name when the program is read into memory. For example:

```
RUN  PR: ABC
 Ø.12345

READY
```

The program is read from the high speed reader and run. It is assigned the name ABC. However, if no filename is specified, the file in memory will have no name. A SAVE command or any other command without a filename cannot be issued without assigning a name to that file (see the RENAME command, Section 2.2.6). For example:

```
RUN PR:
 Ø.12345

READY

SAVE
ILLEGAL FILE NAME

READY
```

A filename must be used in the SAVE command or the program in memory must be given a name.

2.3.2 Program Segmenting, CHAIN Statement

If a user program is too large to be loaded into core and run in one operation, the user can segment the program into two or more separate programs. Each program is assigned a distinct name; control can be transferred from one program to another with the CHAIN statement used as part of a program or in immediate mode.

The immediate mode form of the CHAIN statement is:

CHAIN < *string*>{<*line number*>}

in which <*string*> is the filename specification of the next program segment and <*line number*> specifies the line number in the new program segment at which to begin execution. If no line number is specified, execution begins with the lowest numbered line. For example:

```
CHAIN "PHASE2" 2Ø
```

causes the program file PHASE2 to be executed. Execution begins with line 2Ø in the file PHASE2. The CHAIN command first searches for the file PHASE2.BAC and if that search fails, will search for PHASE2.BAS. A device specification and/or project-programmer number can be included in the filename specification string.

Communication between various program segments can be achieved by means of the user's file area (see the discussion of virtual array files in the BASIC-PLUS Language Manual).

When the CHAIN statement is executed, all currently open files are closed, the new program is loaded, and execution continues. The CHAIN command is similar to the RUN command with the additional capability of specifying a starting line number.

2.3.3 COMPILE Command

Normally, RSTS-11 accepts each line of the user program as it is entered and, if the line is syntactically correct, translates the line into a form understood by the RSTS-11 Run Time System. (The BASIC-PLUS Compiler produces an intermediate code which is then interpreted by the Run Time System.) As the program is edited, only those lines which are changed are recompiled (i.e., translated). When the SAVE command is given, only the source version of the program (i.e., text that is typed in response to the LIST command) is stored in the .BAS file created. In response to the OLD command, BASIC reads the text from the saved file and compiles it in the same manner as when the program is entered from the user keyboard.

Once a program is completely developed and debugged, it may be desirable to avoid the time-consuming practice of compiling the program every time it is brought into memory. For this reason, the COMPILE command has been provided. COMPILE permits the user to save an image of his compiled program, rather than (or in addition to) the source text of the program. This compiled version of the program is stored with the filename extension .BAC and can be read from the system device and executed with a minimum of overhead. (See Section 2.3.1.)

NOTE

> Compiled files have a minimum size requirement of 11 blocks. This size may be greater than necessary to actually store the compiled (or even source) version of a short program. In such cases the user should be aware that he is trading disk space for execution speed.

Due to the transformation that takes place when a program is compiled, a file with the extension .BAC can only be executed; it cannot be edited. Therefore, the user can issue the RUN command with respect to compiled files, but the file cannot be retrieved with the OLD command.

If the current filename (i.e., that which is typed as part of the header listing) is FILEØ1, then the command:

```
COMPILE
```

saves the compiled program in a file named FILEØ1.BAC on the system disk. If another name is desired for the compiled file, it can be specified. For example:

```
COMPILE PROG
```

generates a file PROG.BAC on the system disk. Compiled programs are output only to the system disk.[1] The COMPILE command causes the filename extension .BAC to be appended to the current or specified filename for storage in the user's disk library. Compiled files are stored with a default protection of <124>. The default protection <124> consists of the compiled file protection <64> plus the system-assigned protection code <60>. To obtain any other protection code in addition to the compiled file protection <64>, that code must be explicitly specified as in the following example.

```
COMPILE <4Ø>
```

The above command creates a file on the system disk having the current filename, a .BAC extension, and an assigned protection code of <104>, which consists of the user-specified protection <4Ø> and the compiled file protection <64>. At any time a protection code can be altered with the NAME-AS command or with the rename facility of the PIP system program.[2]

2.4 PROGRAM FILE MANIPULATION

2.4.1 SAVE Command

The SAVE command is used to store BASIC-PLUS source programs on the disk as follows:

```
SAVE
```

The program currently in memory is saved on the system disk under its current filename with the extension .BAS. If a file of the same name exists, the system returns the error message:[3]

```
FILE EXISTS-RENAME/REPLACE
```

---

[1] A privileged user can obtain a copy of a compiled file on another device, using the PIP system program.

[2] On Version 4 (RSTS) systems, the compiled file protection <64> is not used. On Version 5 (RSTS/E) systems, the user can execute the ASSIGN command to alter the protection code assigned by the system (see Section 2.7.3).

[3] Version 4 (RSTS) systems print the message in shorter form: FILE EXISTS-USE REPLACE.

This message is designed to protect the user against inadvertently destroying an existing file. The user can designate a unique name for the file using the SAVE command or can change the current version of the file on the system disk using the REPLACE command.

For example, if the current filename is not the desired name, the format:

```
SAVE FILNAM
```

can be used, which saves the program currently in memory on the system disk under the name FILNAM.BAS. The name of the current source program is not stored in memory with an extension (no extension is printed as part of the current filename when a LIST command is given). The SAVE command appends the .BAS extension when writing the file to a storage device.

In cases where the desired storage device is not the system disk, the format:

```
SAVE dev:
```

or:

```
SAV dev:FILNAM
```

is used where dev: indicates a device designation. The file is stored as the current filename or as FILNAM.BAS on the indicated device. For example:

```
SAVE DTØ:ROPE
```

saves a copy of the program in memory as the file ROPE.BAS on DECtape unit Ø.

The SAVE command is usable only when a source file is currently in memory. When a program is saved, it is still in the computer memory and can be run, changed, or deleted.

To obtain a listing of the current source program on the line printer, the user can type:

```
SAVE LP:
```

To punch a tape of the current source program on the high-speed paper tape punch, the user can type:

```
SAVE PP:
```

No filename specification is needed with an output-only (non-file structured) device.

To punch a tape on the low-speed punch, give a LISTNH command, turning the punch on-line before typing the RETURN key. This is not recommended as the word READY is punched at the end of the tape. Tapes punched on the low-speed punch should be read only through the low-speed reader.

2.4.2 Recalling a Saved Program, OLD Command

To retrieve the source file of a previously saved BASIC-PLUS program, the OLD command is issued as follows:

```
OLD
```

to which the system replies:

```
OLD FILE NAME--
```

The user then types the name of the saved BASIC-PLUS file containing the program. Alternatively, the user can indicate the old filename without prompting, as follows:

```
OLD TAXES
```

which calls the saved file TAXES.BAS from the system disk. If the file is not available on the disk or if it is protected against the user, an appropriate message is printed.

Where no filename is indicated, RSTS-11 looks for the file NONAME (which could have been created by the user or the system, see Section 2.2.1). For example:

```
OLD
OLD FILE NAME--

READY
```

Whatever was stored in the file NONAME.BAS for the current user on the system disk is now in memory and available to the user.

The OLD command can only retrieve BASIC-PLUS source programs, since compiled programs (.BAC files) can be run but not changed. BASIC-PLUS source programs have the default extension BAS but other extensions are allowed. Any program called with the OLD command can be edited by the user at the terminal.

2.4.3 Merging Two Source Programs, APPEND Command[1]

To merge the contents of a previously saved BASIC-PLUS program into a program currently in memory, the APPEND command is issued as follows:

```
APPEND
```

to which the system replies:

```
OLD FILE NAME--
```

The user then types the name of the saved BASIC-PLUS source file to be appended. The file is read into memory and, depending upon the ordering of the source statement line numbers of the appended file and the program currently in memory, its contents are merged into or appended to the current program. If both programs contain an identical line number, the line in memory is replaced by the appended program line. Alternatively, the user can indicate the old file name without prompting, as follows:

```
APPEND RUTINE
```

which merges the contents of the saved file RUTINE.BAS from the system disk into the program currently in memory. If the file is not available on the disk or if it is protected against the user, an appropriate message is printed.

When no file name is indicated, RSTS-11 looks for the file NONAME.BAS, which could have been created by the user or by the system. (See Section 2.2.1 for a description of NONAME.BAS.) For example:

```
APPEND
OLD FILE NAME--

READY
```

The contents of the file NONAME.BAS on the system disk for the current user are merged into or appended to the current program and is available to the user.

The APPEND command can only retrieve BASIC-PLUS source programs (.BAS files), since compiled programs (.BAC files) can be run but not changed. Any file called with the APPEND command is available to the user at the terminal.

[1] The APPEND command is not available on Version 4 (RSTS) systems.

2.4.4 File Protection and Renaming, NAME-AS Statement

The NAME-AS statement can be used in immediate mode to rename and/or assign protection codes to a disk or DECtape file. The format of the command is as follows:

NAME <*string*> AS <*string*>

The specified file, the first <*string*> indicated, is renamed as the second <*string*> indicated.

Where the file resides on a device other than the default device (system disk), the device must be specified in the first string and may optionally be specified in the second string. No filename extension assumptions are made by NAME-AS; the filename extension must be specified in both strings if any extension is present in the filename. For example:

```
NAME "DTØ:OLD.BAS" AS "NEW.BAS"
```

is equivalent to:

```
NAME "DTØ:OLD.BAS" AS "DTØ:NEW.BAS"
```

but the statement:

```
NAME "FILE1.BAS" AS "FILE2"
```

is not advised since FILE2 has no extension and could not subsequently be called into core via the OLD or RUN commands (which require filename extensions).

A file protection code can be specified within typed angle brackets as part of the second string although it is not required. If a new file protection code is specified, it is reflected in the protection assigned to the renamed file. If no new protection code is specified, the old protection code is retained. See the BASIC-PLUS Language Manual for a complete description of protection codes.

```
NAME "FILE.EXT" AS "FILE.EXT <4Ø>"
```

changes only the protection code of the file FILE.EXT stored on the system disk.

```
NAME "DTØ:ABC.BAS" AS "XYZ.BAS"
```

changes the name of the file ABC.BAS on DECtape unit Ø. Since no transfer of the file from one device to another can be performed with the NAME-AS statement, it is not necessary to mention DTØ: twice; that is, the device of the new filename need not be specified. (To transfer a file between devices use the PIP system program.)

```
NAME "NEW" AS "NEW.1"
```

changes only the extension of the disk file NEW (with no extension) to NEW.1.

2.4.5 System Library Files

The system library is conventionally stored under account [1,2] on a RSTS-11 system. The CATALOG$ command can provide each user with a list of those programs available in the system library and their protection codes. Rather than specify the [1,2] account number as part of the file specification each time a system library program is referenced, the user may precede a system library filename with the $ character, which is synonymous with [1,2].

Most system library files are completely write-protected. Unless the file is also read protected, the user can issue OLD or RUN commands with respect to system library files, remembering that the OLD command can only be used on a source file. For example:

```
OLD$ SYSTAT
```

or

```
OLD $SYSTAT
```

(Spaces are not significant in RSTS-11 command strings.)

is equivalent to:

```
OLD [1,2] SYSTAT.BAS
```

and is only legal where SYSTAT is stored as an available source file in the system library. The command:

```
RUN$ SYSTAT
```

will cause the file SYSTAT.BAC to be loaded and executed, if found; and, if SYSTAT.BAC does not exist, the file SYSTAT.BAS is loaded, compiled and executed. (Most system library files are .BAC files.)

2.4.6 Removing a Saved Program, UNSAVE Command

The UNSAVE command is used to remove a file from a storage device. The form:

```
UNSAVE FILNAM.BAC
```

removes the file FILNAM.BAC from the disk. (The system disk is the default device.) If the command is given:

```
UNSAVE FILNAM
```

BASIC attempts to remove the file FILNAM.BAS. Unless a filename extension is specified in the command, an extension of .BAS is assumed.

To indicate an alternate storage device, the form:

```
UNSAVE dev:filename.extension
```

is used where dev: is the device designation. For example:

```
UNSAVE DT1:FLAM
```

removes the file FLAM.BAS from DECtape unit 1 if it is found.

2.4.7 Updating a Saved Program, REPLACE Command

The REPLACE command is used when the user wishes the program in memory to replace a file on the system disk with the same name. The command is of the form:

```
REPLACE
```

or

```
REPLACE FILNAM
```

REPLACE performs the same function as SAVE, but destroys the old copy of the same file, if it exists, without notifying the user. REPLACE appends a .BAS filename extension to the filename already associated with the file or specified in the REPLACE command. Where a filename specification is given, REPLACE replaces the file on the system disk with the name indicated. In this case the name of the current program, as stored in memory, is ignored.

2.5 SYSTEM STATUS REPORTS

2.5.1 LENGTH Command

The LENGTH command returns the length of the user's current program. The user need only type:

LENGTH

and enter the command to the system with the RETURN key. For example:

```
LENGTH
2K OF CORE USED
```

The size of the current program is reported to the next highest 1K increment.

2.5.2 CATALOG Command

The CATALOG or CAT command causes a listing of the current user's disk file directory to be printed on the requesting user terminal.

For example:

```
CAT
LOGIN .BAC        15       124     23-MAR-73 21-MAR-73 05:11 PM
LOGOUT.BAC        15       124     23-MAR-73 21-MAR-73 05:11 PM
SYSTAT.BAC        19       232     23-MAR-73 21-MAR-73 05:11 PM
TTYSET.BAC        19       232     23-MAR-73 21-MAR-73 05:11 PM
PIP   .BAC        23       104     23-MAR-73 21-MAR-73 05:11 PM
EDIT  .BAC        15       104     22-MAR-73 21-MAR-73 05:13 PM
```

name | extension | size in 256-word blocks | protection code | last access date | creation date | creation time

To obtain a catalog of files stored under another user's account number, the command is given as:

CATALOG [1ØØ,1Ø1]

which lists the system disk files owned by user account [1ØØ,1Ø1].

CATALOG$

lists all files in the system library ($ indicates account [1,2], the system library).

To obtain a catalog of files on a device other than the system disk, give the command:

CATALOG dev:

where dev: is a device specification (a user account specification can be included in such a command). For example:

CATALOG DT1:

lists all files on DECtape unit 1 (no account numbers are associated with DECtape files).

CATALOG MT1: [2ØØ,22Ø]

lists all files stored under account [2ØØ,22Ø] on magtape unit 1.

2.6 USING RSTS-11 INPUT/OUTPUT DEVICES

2.6.1 Disable Terminal Echo, TAPE Command

The TAPE command is used to disable the terminal echo feature when reading a paper tape with the low-speed (terminal) reader. The command is given as follows:

TAPE

followed by the RETURN key. The tape is then inserted in the low-speed reader and the reader control switch is set to START.

Prior to giving the TAPE command, the user must set up conditions such that the system expects the tape input. For example, giving the following commands:

```
NEW PROG
TAPE
```

Causes the system to await a source program file to be entered to the system via the terminal tape reader. The terminal echo feature is disabled, so the program is not listed on the terminal as it is read. This allows input to proceed more quickly than typing:

```
NEW PROG
```

and then reading the tape through the low-speed reader. A program listing can be obtained on a line printer or on the terminal at a later time, if necessary.

In TAPE mode, RUBOUT key characters are ignored. RETURN and LINE FEED key characters are transmitted as is, but the LINE FEED or RETURN key characters are not appended since the next character on the input tape is the proper second character.

2.6.2 Enable Terminal Echo, KEY Command

Since no characters input from the terminal keyboard or reader are echoed following the TAPE command, the KEY command is supplied to again enable the terminal echo feature. The user is advised to type the LINE FEED key before issuing the KEY command in case the last line input was not terminated with a carriage return/line feed pair. The command is typed as:

```
KEY
```

and entered to the system with the LINE FEED or ESCAPE key.[1] (Carriage return characters are not treated as delimiters when the terminal is in tape mode.) Following successful entry of the KEY command, characters are again echo-printed at the user terminal.

---

[1] ESCAPE is shown as ALT MODE on some terminals.

2.6.3 Seize Device, ASSIGN Command

The ASSIGN command is used to reserve an I/O device for the use of a single programmer (job number). The command is given in the form:

ASSIGN dev:

where dev: is the device specification (see Table 2-2). If the device is present on the system and available for use, the system returns the message:

READY

If the device is not available for use, the message

DEVICE NOT AVAILABLE

is returned. For example:

```
ASSIGN LP:
READY
ASSIGN PR:
DEVICE NOT AVAILABLE
```

If more than one job is logged into the system under a single account number, only the user (job number) performing an ASSIGN (or DEASSIGN) is affected by that command.

Table 2-2

Device Specifications

Code	Device
PR:	high-speed paper tape reader
PP:	high-speed paper tape punch
CR:	card reader
MTØ: to MT7:	magtape units Ø to 7
MMØ: to MM7:	TMØ2/TU16 magtape units Ø to 7
LPØ: to LP7:	line printer units Ø to 7
DTØ: to DT7:	DECtape units Ø to 7
KB:	current user terminal
KBn:	terminal n in the system
DXØ: to DX7:	floppy disk units Ø to 7

NOTE

The user can reference LPn:, DTn:, DXn:, KBn:, MMn: and MTn: where n is between Ø and the maximum number of such units on the system. LP:, DT:, DX:, MM: and MT: are each the same as specifying unit Ø of the related device.

2.6.4 Release Device, DEASSIGN Command

The DEASSIGN command is used to release the specified device to the device pool within the system (for use by other jobs). If no device is specified, all assigned devices are released from that user (job number). For example:

```
DEASSIGN LP:
```

releases the line printer.

```
DEASSIGN
```

releases all devices previously assigned by that user (under the current job number). If a DEASSIGN command is not given before the user leaves the system, an automatic DEASSIGN is performed when the user gives the BYE command.

2.6.5 Transfer Device, REASSIGN Command[1]

The REASSIGN command transfers control of a device to another job. For example, if DECtape unit 1 is under control of the current job, the following command

```
REASSIGN DT1:8
```

transfers control of the device to job number 8. This command prevents another job from seizing control of the device. An attempt to transfer control of a device to a non-existent job causes the system to generate the ILLEGAL NUMBER (ERR=52) error.[2]

2.7 LOGICAL NAMES FOR DEVICES AND ACCOUNTS[1]

It is often convenient to write programs which reference physical devices by logical names and thus make access to files independent of the physical device on which they reside. The user can subsequently make the necessary associations between logical names and physical devices before the programs are run. To associate a logical name with an assignable physical device, the user types the following form of the ASSIGN command:

```
ASSIGN dev:logical name
```

---

[1] The features described in this section are not available prior to Version 5 (RSTS/E) systems.

[2] On systems prior to V6A, the WHAT error message is printed.

where dev: is the specification of the physical device and logical name is a maximum of six alphanumeric characters. The user can make a maximum of four such assignments for the duration of a job. If the user attempts to make more than four logical assignments, the system prints the ACCOUNT OR DEVICE IN USE error message. The association is unique to the job and is preserved during CHAIN operations. The command does not reserve the device but merely associates the logical and physical names.

If the user makes two logical name assignments for the same device, the system recognizes both logical names as belonging to that device. For example:

```
ASSIGN DT1:A
READY
ASSIGN DT1:B
READY
```

As a result, the system associates both logical names A: and B: with DECtape unit 1.

If the user associates two different devices with the same logical name, the system replaces the former logical assignment with the latter assignment. For example:

```
ASSIGN DT1:A
READY
ASSIGN DT2:A
READY
```

After execution of the above commands, the system associates logical name A: with DECtape unit 2.

If the user associates a device with a valid physical device name, the system recognizes the logical assignment and not the physical assignment. For example:

```
ASSIGN DTØ:DT4
READY
```

The system subsequently associates the physical device designator DT4: with DECtape unit Ø.

To reserve a device for which a logical association exists, the user can type the ASSIGN command with the logical name and a colon as follows.

```
ASSIGN logical name:
```

The system reserves the associated physical device if available. The colon is required. For example, the following commands reserve DECtape unit 1 and associate a logical name to the device.

```
ASSIGN DT1:
READY
ASSIGN DT1:ABC
```

As a result, a statement of the form OPEN "ABC:FILE.EXT" AS FILE 1 in a BASIC-PLUS program subsequently executed attempts to open FILE.EXT on DECtape unit 1. Also, the subsequent use of ABC: in any system command refers to DECtape unit 1. An attempt to refer to a device using an unassigned logical name generates the NOT A VALID DEVICE error (ERR=6).

To cancel the association between a logical name and a physical device, the DEASSIGN command is used as follows.

```
DEASSIGN logical name
```

The command does not release control of the physical device. The system automatically cancels associations between logical names and physical devices when the job is logged out.

To release control of the physical device if a logical name is still in effect, the user can type the following form of the DEASSIGN command.

```
DEASSIGN logical name:
```

The colon is required. The command releases control of the physical device associated with the alphanumeric characters given for logical name. This form of the command does not cancel the association of the logical name with the physical device.

2.7.1 Logical Assignment of a User Account

The ASSIGN command can establish the association between a user account and the commercial at sign (@) character. For example, the command

ASSIGN [1ØØ,1ØØ]

Associates the @ character with the account specified by the project-programmer number [1ØØ,1ØØ]. As a result, subsequent system commands and program statements having the @ character apply to account 1ØØ,1ØØ. An attempt to refer to an account using the @ character with no assignment generages the ILLEGAL FILE NAME error (ERR=2). The command CAT@ prints a directory of account 1ØØ,1ØØ; the command RUN @ DEMO runs the program DEMO from account 1ØØ,1ØØ. In addition, BASIC-PLUS statements such as the following:

OPEN "[1ØØ,1ØØ]FILE.EXT" AS FILE 1

can be shortened in the following manner:

OPEN "@FILE.EXT" AS FILE 1

To cancel the association between the @ character and the account, the DEASSIGN command is used. For example, either the command

DEASSIGN [1ØØ,1ØØ]

or the command

DEASSIGN @

cancels the association between the character @ and account 1ØØ,1ØØ.

2.7.2 Disk Access by Pack Identification Label

Each disk pack or DECpack cartridge on the system has written on it a pack identification label. The user specifies the label in the CCL command, MOUNT (see Section 4.14). This command logically mounts the pack or cartridge on the system. The system monitor stores this label as a device name for the disk pack or DECpack. Subsequently, a user can reference the pack by the pack identification label. For example, if the disk pack MYPACK is logically mounted on drive unit 1, the following command:

CAT MYPACK:

prints a directory of the current account on the disk pack mounted on unit 1. This allows users to refer to a disk pack without knowing the physical disk unit.

To make a logical association between a pack identification label and an alphanumeric string, the user can type the ASSIGN command subsequent to logically mounting the pack. The following sample dialogue shows the procedure.

```
MOUNT DP1:MYPACK
READY
ASSIGN MYPACK:X
READY
```

The logical name X subsequently can be used to refer to the pack logically mounted on RP disk drive unit 1. The logical name X applies only to the current job, whereas the pack identification label is recognized on a system wide basis. The association between the logical name and the pack identification label remains in effect until the pack is logically dismounted, until the DEASSIGN command is executed, or until the job is logged off the system. Subsequent to the assignment of the logical name, a file FILE.EXT on the pack MYPACK logically mounted on RP drive unit 1 can be referenced in any one of the following three ways:

by physical device name,	DP1:FILE.EXT
by pack identification label	MYPACK:FILE.EXT
or by logical device name.	X:FILE.EXT

2.7.3 Changing Default Protection Code

The user can change the default protection code which the system assigns to files created during time sharing. When the user logs into the system, the default protection code is <60>. To change the default, type the ASSIGN command with the new value enclosed in angle brackets. For example,

```
ASSIGN <40>
READY
```

The system sets the default protection code to <40> and assigns that value to any file subsequently created.

2.7.4 Changing the Magtape Labeling Default[1]

The user can change the magtape labeling default when he reserves the magtape unit to the current job. The system-wide labeling default is set by the system manager and remains in effect during the time sharing session unless changed for an individual job. To change the default, type the ASSIGN command with the magtape device designator followed by either .DOS or .ANSI. For example,

```
ASSIGN MTØ:.DOS
READY
```

As a result, the system reserves unit Ø to the current job and treats files on unit Ø as having DOS labels. The DEASSIGN command automatically reverts the unit to the system labeling default. To change the default to ANSI labeling, type a command similar to the following.

```
ASSIGN MTØ:.ANSI
READY
```

If the dot (.) character is omitted, the system assigns the logical name ANSI to the unit.

[1] This feature is not available prior to Version 6A (RSTS/E) systems.

2.8 SCALED ARITHMETIC, SCALE COMMAND[1]

The scaled arithmetic feature is available to overcome accumulated errors in fractional computations performed on systems using the four word floating point format. The feature enables the system to maintain decimal accuracy of fractional computations to a given number of places determined by a scale factor. The user can control the scale factor according to guidelines described in this section.

Users on a system with the double-precision, four word floating-point format can optionally execute the SCALE command to control the scale factor. An attempt to use the SCALE command on systems with the single precision floating point format results in the system printing the MISSING SPECIAL FEATURE error message. Scaled arithmetic is described in Section 6.8 of the BASIC-PLUS Language Manual.

To specify the scale factor to be used, type the SCALE command with a decimal integer between Ø and 6. For example,

```
SCALE 2
READY
```

The SCALE 2 command sets the current scale factor to 2. Subsequently, all programs compiled for that job have a scale factor of 2. If an invalid scale factor is typed with the command, the system prints the SYNTAX ERROR message.

NOTE

SCALE is solely a system command and cannot be executed as a BASIC-PLUS statement. Also, a program cannot refer to or modify the scale factor.

Typing a value of Ø with the SCALE command disables the scaled arithmetic feature. For example,

```
SCALE Ø
READY
```

The SCALE Ø command sets the scale factor to Ø. Programs compiled for that job subsequently treat all floating point calculations in the standard fashion. When a user logs a job onto the system, the initial scale factor is Ø.

[1]This feature is not available prior to Version 5 (RSTS/E) systems.

To determine the current scale value, the user types the SCALE command without a value. For example,

```
SCALE
6
READY
```

The system prints the current value (6) and the READY message. If the program in memory has a value set other than the current value, the system prints two values. For example,

```
SCALE
6,2
READY
```

The first value (6) indicates the current value for the job, and the second value (2) is the one set for the program currently in memory. If the scaled arithmetic option is currently disabled, the system prints a zero as the first value.

When a user loads source statements into memory, whether by an OLD command, a RUN command to a BAS file, or by entering statements after a NEW command, the system sets the scale factor for the program in memory to the current scale factor.

```
SCALE 4

READY

OLD TEST

READY

LISTNH
10      A$ = "##.#####"
20      X  = .12345
30      PRINT USING A$,X
40      END

READY

RUNNH
 0.12340

READY
```

If he executes a RUN command for the program in memory, the system performs floating point calculations using the scale factor of the program currently in memory. Similarly, the system uses the scale factor of the program in memory when executing immediate mode statements.

After loading source statements into memory, the user cannot change the scale factor for the program in memory. This action prevents the user from possibly executing floating point calculations for a program, parts of which the system assigned a different scale value.

An attempt to execute such a program or source statement causes the SCALE FACTOR INTERLOCK warning message. For example,

```
SCALE
4

READY

OLD TEST

READY

SCALE 6

READY

RUNNH
SCALE FACTOR INTERLOCK
 0.12340

READY
```

The program executes using the scale factor of the program in memory (4) rather than the current scale factor (6). To execute such a program with a changed scale value, the user can type a SAVE or REPLACE command followed by an OLD or RUN command for the related file. For example,

```
REPLACE TEST

READY

OLD TEST

READY

RUNNH
 0.12345

READY

SCALE
6

READY
```

The system consequently executes the program with the changed scale factor 6.

The following example illustrates the effect of different scale values for a sample program TEST.BAS

```
SCALE 2

READY

OLD TEST

READY

RUNNH
 0.12000

READY

SCALE 4

READY

OLD TEST

READY

RUNNH
 0.12340

READY
```

The system loads the source file using the current scale factor 2. When executed, the program generates results to two decimal places, and truncates the remaining places. If the user changes the current scale factor and loads the same program, the system executes the program with changed scale factor and truncates the remaining places.

If the user stores the compiled file and later runs it with a different scale value in effect, the following occurs.

```
SCALE 6

READY

OLD TEST

READY

COMPILE TEST

READY

SCALE 4

READY
```

```
RUN TEST
 0.12345

READY

SCALE
4,6

READY
```

The system loads the program TEST and executes it with the scale value (6) set when the user compiled it. The system does not use the current value (4) but rather the value of the compiled program (6). This action occurs whether the program runs by a RUN command or by a CHAIN command.

If a compiled program is currently in memory, the user cannot change its scale factor and run that program. Since it is impossible to alter the scale factor of a compiled program except by compiling the program again, the system generates the SCALE FACTOR INTERLOCK warning message. For example,

```
SCALE
4,6

READY

RUNNH
SCALE FACTOR INTERLOCK
 0.12345

READY
```

The program executes with the scale factor of the compiled program (6).

CHAPTER 3
SPECIAL CONTROL CHARACTERS

3.1 RETURN KEY

Typing the RETURN key echoes as a carriage return/line feed operation on the terminal, as long as the terminal is not in tape mode. The RETURN key is normally used to terminate a line and enter that line to the system. In tape mode (following entry of the TAPE command) all carriage returns are ignored.

3.2 ESCAPE OR ALT MODE KEY

The ESCAPE key, like the RETURN key, is used to terminate the current line and causes the line to be entered to the system. However, the ESCAPE key echoes on the terminal paper as a $ character and does not perform a carriage return/line feed. ESCAPE is used to enter the KEY command to the system.

On some terminals the ESCAPE key is replaced by the ALT MODE key, which performs the same functions.

3.3 LINE FEED KEY

The LINE FEED key is generally used to continue the current logical line of input on an additional physical line. The LINE FEED key does not echo on the terminal paper but does perform a carriage return/line feed operation when used with the RSTS-11 system. Typing the LINE FEED key automatically generates a physical, but not logical, carriage return/line feed sequence; so the LINE FEED is not used to terminate or enter lines to the system (except for the KEY command).

LINE FEEDs produce errors in programs if included in constants (including string constants), verbs, or user-specified names for variables or functions.

3.4 RUBOUT KEY

The RUBOUT key is used as an eraser for the current line. If typed in tape mode, the RUBOUT key is ignored; otherwise, it causes the last character typed to be deleted. The erased characters are echoed on the terminal paper between backslashes. For example:

```
1Ø LEF X=X*X
```

could be corrected by typing the RUBOUT key 7 times (to remove the F) and typing the remainder of the line correctly. The line would look as follows on the terminal paper:

```
10 LEF X=X*X\X*X=X F\T X=X*X
```

and would appear to the system as:

```
10 LET X=X*X
```

In cases where the mistake is toward the beginning of a line, it may be easier to simply retype the entire line. For example:

```
10 LEF X=X*X
10 LET X=X*X
```

Once the second line is entered to the system, the first line number 10 is deleted.

RUBOUT can be used on any input line to the RSTS-11 system, including commands, as long as the line has not been entered to the system with RETURN, LINE FEED or ESCAPE.

3.5 CTRL/C

Typing a CTRL/C (hold down the CTRL key and type the C key, release both) causes RSTS-11 to print READY and return to command mode where commands can be given or editing done. CTRL/C stops whatever RSTS-11 was doing at the time (execution or output) and returns control of the system to the user.

Note that CTRL/C interrupts processing. For example, if CTRL/C is used after the REPLACE command is issued and before the READY reply is received, the file is not replaced in its entirety and is not closed. Since the file has not been closed, it cannot be accessed later. Similarly, if the OLD command is issued and a list of error messages is being printed, the CTRL/C key should not be used since the current program is only half compiled at that point.

3.6 CTRL/U

The CTRL/U combination deletes the current input line. This is useful when a long command has been typed and is seen to be incorrect. Rather than use the RUBOUT key repeatedly, CTRL/U deletes the entire line. This feature can be used when typing either commands or statements.

CTRL/U deletes the entire current physical line. For example:

```
10 PRINT "ALPHABET"^U
LISTNH

READY
```

In typing line 1Ø a mistake was made and the line deleted with CTRL/U. Notice that the line does not appear in the program listing following LISTNH. In typing line 2Ø, below, the LINE FEED key is used to continue line 2Ø onto a second line. A physical line has been terminated.

```
20 LET M =
278.12^U

SYNTAX ERROR AT LINE 20

READY
```

The logical statement at line 2Ø, however is continued onto the following line and its deletion with a CTRL/U causes the statement at line 2Ø to be entered as:

```
LISTNH
?20 LET M =


READY
```

which is syntactially incorrect. Had the last line been terminated with the RETURN key it would be entered to the system as:

```
20 LET M =
278.12

READY
```

which would be accepted as equivalent to:

```
20 LET M = 278.12
```

3.7 CTRL/O

The CTRL/O combination suppresses output to the terminal until the next time CTRL/O is typed. When a program produces a large amount of output (usually tabular form), the user may not wish to wait for the printing of the complete information. CTRL/O enables the user to monitor the output while not stopping it completely. Typing CTRL/O while occurring still allows the computer to output the data, but the terminal does not print it. This speeds up the output process, since terminals are normally slow devices. The second time CTRL/O is typed, the output is again printed at the terminal.

CTRL/C, on the other hand, completely terminates program output. Think of CTRL/O as a switch, the first setting of which creates a condition and the second setting releases the condition.

3.8 TAB CHARACTER

The TAB character or CTRL/I combination allows the user to insert a tabular format into his typed material. When entering a program to the system, the TAB character allows formatting such as shown in Figure 3-1. The RSTS-11 editor considers each Teletype line to contain eight tab stops, eight spaces apart, across the line. Typing the TAB character causes the printer head to move to the next tab stop on the current line.

If using a model ASR-33 or KSR-33 Teletype, typing the TAB character echoes the appropriate number of spaces to reach the next tab stop. The model 35 Teletypes have hardware tab stops. (See the description of the TTYSET system program for a means to enable or disable hardware tabstops.)

```
20 IF A>B THEN
        IF B>C THEN PRINT "A>B>C"
               ELSE IF C>A
                       THEN PRINT "C>A>B"
                       ELSE PRINT "A>C>B"
        ELSE IF A>C THEN PRINT "B>A>C"
               ELSE IF B>C
                       THEN PRINT "B>C>A"
                       ELSE PRINT "C>B>A"
```

Figure 3-1 TAB Character Example

3.9 CTRL/Z

The CTRL/Z combination is used to mark the end of a data file. When data is input from a file, the CTRL/Z character marks the end of recorded data. The message "END OF FILE ON DEVICE" is printed by the system when a ↑Z is detected unless an ON ERROR GOTO statement is used to enable a BASIC-PLUS routine to handle the error (ERR = 11).

3.10 CTRL/S[1]

The CTRL/S combination is used to suspend temporarily output to an alphanumeric display terminal. This feature is used to examine the lines currently displayed on the screen before they are replaced by the output of additional lines on the screen. Output can be resumed at the next character by typing the CTRL/Q combination. This feature is usable only if the terminal has been initially defined with the STALL characteristic (see Section 4.5, TTYSET Program).

[1]This feature is not available before Version 5 (RSTS/E) systems.

CHAPTER 4

RSTS-11 SYSTEM PROGRAMS

In addition to the core-resident BASIC Compiler, there are several disk-resident system programs available on RSTS-11. The list of programs includes the following:

LOGIN	allows users to enter the system.
LOGOUT	allows users to leave the system.
SYSTAT	provides system status summaries.
PIP	moves data from one peripheral device to another.
TTYSET	allows the user to specify functional characteristics of the user terminal.
QUOLST	allows the user to determine his current disk quota and the amount of file space on each system device.
MONEY	allows the user to determine the amount of time charged to his account number.
GRIPE	allows the user to log complaints or comments about the system.
EDIT	provides program and text editing compatible with DOS/BATCH-11
BACKUP	transfers files to and recalls files from a reserve storage medium.
QUE[1]	enters a user's job in a system queue file for subsequent processing by a spooling program.
RUNOFF[1]	generates a formatted listing from an ASCII file containing RUNOFF commands and user's text. See the RSTS/E RUNOFF User's Guide for operating information.
BATCH[1]	executes an unattended job by processing user specified, standard, control language statements from a device or file queued for the batch device.
DIRECT[1]	quickly generates a listing of disk directory.
UMOUNT[1]	mounts or dismounts a user's private disk.
FILCOM[1]	compares two files and reports differences found.
INUSE[1]	prints INUSE at a terminal.
COPY[1]	performs fast copies of DECtapes, magtapes and RK cartridge disks.
COBOL[2]	loads and runs the PDP-11 COBOL Compiler on systems with the auxiliary run time system RTSLIB. See the PDP-11 COBOL User's Guide for operating information.

[1] These features are not available prior to Version 5B (RSTS/E) systems.
[2] These features are optional and are not available prior to Version 6A (RSTS/E) systems.

SORT11[1]	loads and runs the PDP-11 SORT11 program on systems with the auxiliary run time system RTSLIB. See the PDP-11 SORT Reference Manual for operating information.
PIP[1] (extended)	the extended Peripheral Interchange Program which requires 16K words of user space.

The functions and techniques for using each program are described in the following sections.

This Chapter was designed in a modular format so that the system manager could remove or add sections depending upon the needs of the individual installation. (The Table of Contents reflects the material supplied by DEC.)

The user can run some system programs by typing a unique system command called a Concise Command Language (CCL) command. Use of CCL commands requires that the concise command language option be included on the system at system generation time.[2] If CCL is included on the system, up to 20 such commands are available.

CCL commands allow a user to run a system program and to specify a single command for the program to execute. The user types the CCL command and the program command on one line and enters it to the system. The system loads the program into the user's job area and writes the program command to the core common area. This operation destroys the current contents of the user's job area and is similar to that performed by the HELLO and ATTACH commands which load and run LOGIN. The program runs, reads the command from the core common area, and executes it. If an error is encountered, the program prints a related message and terminates.

Although CCL commands are installation dependent, DEC provides a table of CCL commands available on most systems. Shown below are some of the CCL commands.

CCL Command	Associated Program
PIP	PIP
HELP	PIP
QUE	QUE
EDIT	EDIT
CREATE	EDIT

[1]These features are optional and are not available prior to Version 6A (RSTS/E) systems.
[2]This feature is not available prior to Version 5 (RSTS/E) systems.

CCL Command	Associated Program
DIR	DIRECT
SYS	SYSTAT
SET	TTYSET
MOUNT	UMOUNT
DISMOUNT	UMOUNT
CBL[1]	COBOL
COBOL[1]	COBOL
SRT[1]	SORT11
SORT[1]	SORT11

[1] These commands were not available prior to Version 6A systems.

THIS PAGE PURPOSELY LEFT BLANK.

4.1 LOGIN PROGRAM

The LOGIN system program runs from either a logged in or a logged out terminal. It activates a job at a terminal, attaches a detached job[1] to a terminal, or runs designated system programs from a logged out terminal.

4.1.1 Running LOGIN From a Logged Out Terminal

As described in Chapter 2, the LOGIN system program runs when either HELLO, LOG[2], LOGIN[2], ATTACH[2], ATT[2] or I is typed at a terminal connected to the RSTS-11 system. This section describes more fully the actions which occur when LOGIN runs at a logged out terminal.

When a terminal is connected to the RSTS-11 system by a dial up connection, the automatic answering signal causes the system monitor to insert an I in the input buffer for that terminal. This action simulates an I being typed at a terminal directly connected to the system. The description of the resultant actions in this section applies to the cases of a terminal directly connected to the system and of a terminal connected by a dial up line.

When a user enters typed characters to the system from a terminal directly connected to but not logged into the system, the monitor runs the LOGIN system program which checks the characters for a valid command. If only the RETURN key is typed, the system takes no action. A user entering either SYS or SET commands causes LOGIN to chain to line 32000 of the SYSTAT or TTYSET system programs, respectively. The system manager can alter the LOGIN program to run other programs in the same manner.

---

[1] A job becomes detached because the connection of a dial up line is broken or a privileged job executed the SYS system function to detach the job from the terminal.

[2] These commands are not available prior to Version 5 (RSTS/E) systems.

If the user types HELLO, LOGIN, LOG, ATTACH, ATT, or I, LOGIN prints the system identification line as in the following sample printout.

```
HELLO

RSTS V06A-01 SYSTEM #880  JOB 12  KB16  03-JUN-75  08:37 AM
#120,80
```

The line contains the system name and version number, the local installation name, the job number activated, the keyboard number of the terminal, and the current system date and time. The pound sign (#) character printed by LOGIN on the following line requests the user to type his account number.

The user types the project and programmer numbers separated by either a comma or a slash and terminated with the RETURN key. (Typing the slash as a separator inhibits printing of any system notice messages.) The user can specify the project and programmer numbers on the same line as the HELLO, LOGIN, LOG or I commands as shown in the following sample dialog.

```
HELLO 120,80
PASSWORD:
```

When an account number is included in the command, LOGIN does not print the # character but immediately prompts the user to enter the password. LOGIN disables echo printing at the terminal when the password is typed.

If either the account does not exist or the password does not match, LOGIN prints the INVALID ENTRY - TRY AGAIN message and the # prompting character. The user can try the sequence to a maximum of five times. LOGIN allows the user 30 seconds in which to type an entry. After the fifth invalid entry, LOGIN prints the ACCESS DENIED message and frees the job for other usage.

A valid entry causes LOGIN to check for any other jobs which may be running on the system under the same account number. If other jobs are running and none are detached, LOGIN reports how many such jobs by printing a message similar to the following sample and prints the system notice messages.

```
1 OTHER USER(S) ARE LOGGED IN UNDER THIS ACCOUNT
```

If any jobs are running detached under the current account, LOGIN instead reports the number of each such job and requests the user to type the number of the job to be attached to the terminal. The following sample printout shows the procedure.

```
JOB(S) 10 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO? 10
ATTACHING TO JOB 10

READY
```

To attach a job to the terminal, simply type its number in response to the query. LOGIN prints the ATTACHING TO message and attempts to attach the specified job to the current terminal. When the job is attached, the current job is freed for other usage and the attached job runs at the terminal.

To continue running the current job, the user simply types the RETURN key in response to the query. LOGIN subsequently prints the message concerning other jobs running under the same account and prints the system notice messages, if any.

```
JOB(S) 10 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO?
1 OTHER USER(S) ARE LOGGED IN UNDER THIS ACCOUNT


READY
```

System notices convey to the user information which the system manager places in the file NOTICE.TXT in the system library. If the file does not exist, LOGIN proceeds. If the user's job is running under a non-privileged account, LOGIN sets its maximum job size to 8K words. The LOGIN.BAS file can be modified so that LOGIN sets a different job maximum.[1] The maximum size of a job running under a privileged account is normally set to 16K words. LOGIN prints the READY message and exits to the system monitor which clears the LOGIN program from memory.

The complete sequence to log a job into the system when other jobs are running detached is shown below.

```
HELLO 1/210
PASSWORD:
JOB(S) 10 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO?
1 OTHER USER(S) ARE LOGGED IN UNDER THIS ACCOUNT


READY
```

[1] On Version 4 (RSTS) systems, the maximum job size is determined by the amount of free core on the system, and cannot be specified by LOGIN.

The complete sequence to attach another job to the terminal when logging into the system is shown in the following sample printout.

```
HELLO

RSTS V06A-01 SYSTEM #880  JOB 12  KB16  03-JUN-75  08:49 AM
#1/210
PASSWORD:
JOB(S) 10 11 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO? 10
ATTACHING TO JOB 10

READY
```

To attach a job to a terminal when the job number is known, the user can type the ATTACH or ATT command as follows:

```
ATTACH

RSTS V06A-01 SYSTEM #880  JOB 8  KB16  03-JUN-75  08:51 AM
JOB NUMBER TO ATTACH TO? 8
JOB NOT DETACHED - ACCESS DENIED
```

LOGIN runs and prints the system identification line and, on the next line, the JOB NUMBER TO ATTACH TO query. The user must type the number of the detached job. If the job is not detached or does not exist, LOGIN prints an appropriate message followed by ACCESS DENIED. The used must type another command to log into the system.

If the job is detached, LOGIN prompts the user for the password of the account under which the detached job is running. After the user enters the password, LOGIN prints the ATTACHING TO JOB x message and attempts to attach the specified job to the terminal. An incorrect password causes LOGIN to print the FAILURE TO ATTACH TO JOB x message and to terminate as shown in the following sample printout.

```
ATTACH

RSTS V06A-01 SYSTEM #880  JOB 10  KB16  03-JUN-75  08:52 AM
JOB NUMBER TO ATTACH TO? 4
PASSWORD:
ATTACHING TO JOB 4
FAILURE TO ATTACH TO JOB 4
```

The user must try again. If the system successfully attaches the job to the terminal, the terminal becomes the console terminal of the job. Further terminal output is under programmed rather than system control.

To omit the printing of the identification and the query lines, simply type the job number on the same line as the ATTACH or ATT command as follows.

```
ATT 7
PASSWORD:
ATTACHING TO JOB 7

READY
```

The READY message indicates that the attached job is at the system monitor level.

4.1.2 Running LOGIN at a Logged in Terminal

If the user types the HELLO or the ATTACH[1] command at a terminal already logged into the system, the LOGIN system program is loaded into the user's job area and is started. The previous contents of the user's area are destroyed. LOGIN prints the system identification line with one additional item inserted. Between the job number and the keyboard number printed on the line, LOGIN inserts the project-programmer numbers of the account under which the current job is running. Typing the LOGIN, LOG, ATT or I command at a terminal already logged into the system causes the system monitor to print the WHAT? error message and the READY message.

LOGIN determines if any other jobs are running under the same account and prints the message informing the user of the quantity of those jobs. The following sample dialog shows the procedure.

```
READY

HELLO

RSTS V06A-01 SYSTEM #880  JOB 7  [120,80]  KB16  03-JUN-75  09:31 AM
1 OTHER USER(S) ARE LOGGED IN UNDER THIS ACCOUNT


READY
```

[1] The ATTACH command is not available prior to Version 5 (RSTS/E) systems.

If any such jobs are running detached, LOGIN instead prints the message informing the user of the number of each such job and, on the following line, prints the ATTACH TO query as follows.

```
READY
HELLO

RSTS V06A-01 SYSTEM #880  JOB 12  [1,210]  KB16  03-JUN-75  08:58 AM
JOB(S) 10 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO? 22
NO JOB BY THAT NUMBER - TRY AGAIN
JOB NUMBER TO ATTACH TO?
```

To attach one of the jobs to the terminal, the user types one of the job numbers reported in the message. LOGIN determines whether the job is nonexistent or whether it is already attached to another terminal. In either case, the program prints an appropriate error message saying try again and subsequently reprints the ATTACH TO query.

To continue running the current job, type the RETURN key in response to the ATTACH TO query. As a result, LOGIN prints the information message telling how many other jobs are running under the same account and prints the READY message. The system clears the LOGIN program out of memory.

```
JOB NUMBER TO ATTACH TO?
1 OTHER USER(S) ARE LOGGED IN UNDER THIS ACCOUNT

READY
```

When the user responds to the ATTACH TO query by typing one of the job numbers reported in the message, the program proceeds as shown in the following sample printout.

```
HELLO

RSTS V06A-01 SYSTEM #880  JOB 12  [1,210]  KB16  03-JUN-75  09:01 AM
JOB(S) 10 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO? 10
ATTACHING TO JOB 10

READY
```

The READY message indicates that the new job is at system command level.

To attach to a job known to be running detached under the same account, the user can type the job number on the same line as the ATTACH command. The LOGIN program determines if the job specified exists and is detached. If not, it prints an appropriate error message and the ATTACH TO query. The user can type another job number or the RETURN key. If the job exists and is detached, LOGIN compares the account numbers under which both the current job and the detached job are running. If the accounts are different, the program prompts the user for the password of the account under which the detached job is running. The following sample printout shows the procedure.

```
ATTACH 22
NO JOB BY THAT NUMBER - TRY AGAIN
JOB NUMBER TO ATTACH TO? 10
PASSWORD:
ATTACHING TO JOB 10
FAILURE TO ATTACH TO JOB 10


READY
```

After the user enters the password, the program prints the ATTACHING TO message and attempts to attach the detached job to the terminal. If the password is not valid, the program prints the FAILURE TO ATTACH and the READY messages and exits to the monitor, which clears the LOGIN program from the user's job area.

When the account numbers of the two jobs are the same, LOGIN omits the PASSWORD prompt message and attaches the job as shown below.

```
ATTACH 10
ATTACHING TO JOB 10

READY
```

The READY message indicates that the job is at the system monitor level.

To change accounts without logging off the system, simply type the HELLO command followed by the account number.[1] For example:

```
READY

HELLO

RSTS V06A-01 SYSTEM #880  JOB 10  [120,80]  KB16  03-JUN-75  09:09 AM

READY

HELLO 120/81
PASSWORD:

READY

HELLO

RSTS V06A-01 SYSTEM #880  JOB 10  [120,81]  KB16  03-JUN-75  09:09 AM

READY
```

To have the system print the system notice message, merely replace the / character in the account number with a comma.

[1] This feature is available only on Version 5 (RSTS/E) systems.

4.1.3 Running Other Programs from a Logged Out Terminal

Certain commands typed at a logged out terminal cause LOGIN to chain to line 32000 of another program in the system library. The system manager can modify the LOGIN program to recognize other commands and to chain to line 32000 of a program stored in the system library. The commands which the unmodified LOGIN recognizes are SYS and SET which respectively cause the SYSTAT and TTYSET system programs to run.

For example, it is convenient to determine job status without logging a job into the system. The following printout shows the procedure.

```
SYS 11
 11     1,210      KB17    NONAME     2K     KB                    0.1
BYE
```

LOGIN runs and recognizes the SYS command of the SYSTAT system program. LOGIN writes the option given in the command in the core common area and chains to the SYSTAT program at line 32000. SYSTAT reads the option from the core common area and prints the appropriate report, after which it chains back to LOGIN. LOGIN prints the BYE message and exits to the system monitor which clears the contents of memory.

4.2 LOGOUT PROGRAM

LOGOUT is called when the user has completed all processing and is ready to leave the terminal. The LOGOUT program is started when the BYE command is typed at a user terminal logged into the RSTS-11 system. LOGOUT checks the current user's disk quota to ensure that the user does not log out of the system with more than the acceptable amount of disk storage being used for his files. If the user's disk files are within the acceptable disk quota size, LOGOUT disconnects the terminal from the system, removes the current job number from the list of active jobs and prints some information on the duration of the current job.

In response to the BYE command, LOGOUT prints:

CONFIRM:

The user can type any of the responses shown in Table 4-1.

Table 4-1
LOGOUT CONFIRM: Responses

CONFIRM: Response	Meaning
Y	The system performs the checks described above. If successful, the LOGOUT messages are printed. If not successful, an error message is printed and the user must delete some files.
N CTRL/C	These responses indicate that the user does not want to log out of the system. The LOGOUT procedure is terminated without logging the user off the system and the system prints the READY message.
?	Causes LOGOUT to print an explanation of the acceptable responses to CONFIRM:
RETURN key	Causes LOGOUT to print a message instructing the user to type ? to obtain a description of logout procedures.
I	Causes LOGOUT to enter individual file deletion mode.
Other	Same as RETURN key.
F	Causes a fast logout procedure if user's disk storage space is within acceptable limits.

In individual deletion mode, LOGOUT prints the name, size, protection code, and creation date of each file stored under the current user account number on the system disk. This

information is followed by a ? after which the system awaits a response from the user which can be:

File Deletion Mode Response	Meaning
RETURN key	Save the file just listed.
K	Delete (kill) the file just listed.

An example of a LOGOUT sequence is shown below:

```
BYE Y
DISK QUOTA OF 400 EXCEEDED BY 50 BLOCKS
SOME FILE(S) MUST BE DELETED BEFORE LOGGING OUT
TYPE '?' FOR HELP
CONFIRM: I
A       .BAS        150      60      15-AUG-74   ?
C       .BAS        150      60      15-AUG-74   ?
B       .BAS        150      60      15-AUG-74   ? K
TYPE '?' FOR HELP
CONFIRM: Y
SAVED ALL DISK FILES; 300 BLOCKS IN USE, 100 FREE
JOB 18 USER 120,80 LOGGED OFF KB15 AT 15-AUG-74 04:06 PM
SYSTEM RSTS V05B-24 SYSTEM #880
RUN TIME WAS 1.6 SECONDS
ELAPSED TIME WAS 1 MINUTE, 31 SECONDS
GOOD AFTERNOON
```

LOGOUT statistics for a job which was detached at any time include accumulated run time and elapsed time statistics.

The user can omit the CONFIRM: message by typing the BYE command and the response to the CONFIRM: message[1]. For example, to perform a fast logout, type the following.

```
BYE F
```

The LOGOUT program runs and performs the fast logout procedure by printing a series of LINE FEED characters instead of printing the final accounting information. If the user job exceeds the acceptable limit for disk storage, LOGOUT prints the QUOTA EXCEEDED message and the CONFIRM: message to allow the user to delete some files before logging out.

[1]This feature is not available prior to Version 5 (RSTS/E) systems.

4.3 SYSTAT PROGRAM

The SYSTAT program provides current system information in the areas of job, device, disk, and buffer status. SYSTAT can be called by a user logged into the system or from a terminal which is on-line but not logged into the system.

To start SYSTAT while logged into the system, type:

```
RUN $SYSTAT
```

If not logged into the system, type:

```
SYS
```

If the user is already logged in, the system responds by printing:

```
OUTPUT STATUS TO?
```

at which point the user can indicate any RSTS-11 device or a filename specification for the status report output. Possible replies by a user logged into the system are described below:

SYSTAT Output Response	Meaning
LP:	send status report to the line printer if only one line printer is on the system or to line printer unit Ø if multiple line printers are on the system.
LPn:	send status report to line printer unit n if that printer is not currently in use.
KB:	send status report to the user terminal. (The RETURN key is equivalent to responding KB:)
KBn:	send status report to user terminal n in the system if that terminal is on-line and not currently in use.
PP:	send status report to the high-speed paper tape punch.
dev:filename.ext	send the status report to the file specified. The default device is the system device. No extension is appended unless specified by the user.

If SYSTAT is run by a user not logged into the system, the report is always sent to the user terminal requesting the report.

Following the device or filename specification, the user can specify one of the options in Table 4-2 to obtain a partial system status report. The option specifications are preceded by a slash if the user is logged into the system. The options can be typed following the SYS command if the user is not logged into the system.

Table 4-2

SYSTAT Options

SYSTAT Option Specification	Meaning
null	Report complete system status to include job, run time system, busy device, disk structure and free buffer status statistics.
/R	report only run time system statistics[1]
/n	report status for job n only
/S	report only job status
/B	report only busy device status
/D	report only disk status
/F	report only free buffer status
/Kn	report only the status of terminal n in the system
/DET	report only the status of detached jobs[2]
/n,m	report only the status of any jobs active under account [n,m]
/Ø,Ø	report only the status of jobs not logged into the system

The options S, B, D, and F can be specified as separate options or in any combination.

The following examples are performed on a terminal logged into the system:

RUN $SYSTAT
OUTPUT STATUS TO? STAT — creates complete system status report in the file STAT under the current account in the public structure.

RUN$ SYSTAT
OUTPUT STATUS TO? LP: /3 — causes output of a status report for job 3 to the line printer.

RUN$ SYSTAT
OUTPUT STATUS TO? /D — causes output of disk status report to the user terminal.

RUN$ SYSTAT
OUTPUT STATUS TO? /SF — causes output of job and free buffer status to the user terminal.

[1] This feature is not available before Version V6A systems.

[2] Normally, a job is detached only by use of a certain system function or because of a dataphone disconnect affecting a remote user.

The following examples are performed on a terminal not logged into the system:

SYS	causes output of complete system status report to the terminal.
SYS /D	causes output of the disk status report to the terminal.
SYS /BF	causes output of the busy device and free buffer status reports to the terminal.
SYS /5	causes output of a status report for job 5 to the terminal.
SYS A/B	causes the FATAL SYSTEM I/O FAILURE error since SYSTAT cannot create a file for a logged out job.

A complete system status report is shown below:[1]

```
RUN $SYSTAT
OUTPUT STATUS TO?

RSTS V06A-01 SYSTEM #880 STATUS ON 02-JUN-75 AT 04:15 PM UP: 25:19:34

JOB     WHO       WHERE     WHAT    SIZE     STATE      RUN-TIME
 1     [OPR]       DET     ERRCPY     4K     SL SW          27.9
 2     [OPR]       DET     QUEMAN    16K     SL           3:01.2
 3     [OPR]       DET     SPOOL     10K     SL SW       13:31.1
 4     [OPR]       DET     SPOOL     10K     SL SW           5.0
 5     [OPR]       DET     BATCH     15K     SL SW        1:33.7
 6     [OPR]       DET     BATCH     15K     SL SW          56.3
 7       2,100     DET     VT5DPY    14K     SL SW       25:47.5
 8    220,60       KB24    NONAME     2K     KB SW        1:39.3
 9       1,206     KB30    TECO      14K     KB SW        9:51.7
 10    [SELF]      KB16    SYSTAT     8K     RN              3.5
 11   253,10       KB8     TECO       3K     KB SW          35.0
 12   140,21       KB32    NONAME     2K     KB SW          33.4
 13    [OPR]       KB14    QUE       11K     KB SW           4.6
 14    [OPR]       KB11    MAC       18K     KB           1:39.1
 16   250,81       KB15    SORT11     8K     KB SW        6:38.1
 17   240,1        KB17    TECO       5K     KB SW          46.5

RUN-TIME SYSTEMS:
 NAME    SIZE     USERS    COMMENTS
BASIC     15K       10      READ ONLY
RTSLIB     4K        1      READ ONLY, TEMPORARY

BUSY DEVICES:
DEVICE   JOB      WHY
KB12      7       INIT
LP0       3       AS
```

[1] On Version 4A-12 (RSTS) systems, SYSTAT prints a count of the catastrophic errors in the buffer status information. On RSTS/E systems before Version 6A systems, SYSTAT does not print run time system or priority information.

```
DISK STRUCTURE:
DISK     OPEN      FREE    CLUSTER ERRORS  COMMENTS
DB0       7       23648     4        35      PUBLIC
DB1       10      38036     4        0       PUBLIC

SMALL    LARGE    HUNG TTY'S
 466      1         0

READY
```

The job status information includes a list of all active jobs by job number, the account number under which each job runs, the keyboard involved, the program name and size, the job state, the total amount of central processor run time exhausted, and the job priority. In the WHO column, SYSTAT substitutes OPR for the project-programmer number to denote an operator account. An operator job has a project number of 1 and a programmer number between 1 and 2ØØ. In the WHERE column, DET appears in place of the keyboard number for job which run detached from a keyboard. The SIZE column shows two numbers separated by a slant character. The first number is the current size in words; the second number is the size to which the job can expand. The STATE column contains an abbreviation (see Table 4-3) telling the condition the job is currently in. The RUN-TIME column gives hours, minutes, seconds, and tenths of seconds of central processor time the job has consumed. The PRIORITY column (printed only if job is privileged), gives the number which the system assigns to determine the order in which to run jobs. Most jobs run at -8 priority; system jobs run at special priorities of Ø or higher.

The run time system information gives the name of each module, its size in words, the number of user jobs currently executing under its control, and comments regarding its status.

The device status information reports devices which are busy, having been assigned or opened by a specific user. Items reported are the device specification, the job owning that device, and the condition of the device.

The disk status information describes each disk (public and private) currently mounted. Items reported are: disk name (device specification), number of open files, number of free 512-byte blocks, pack cluster size, disk hardware error count, and the public/private and locked/unlocked status of the device.

The buffer status provides information on the number of small (16-word) and large (256-word) buffers currently not in use, and a count of the number of times a hung terminal was found. A hung terminal is one which fails to respond to character transmission within a given time period. The system attempts to unhang the terminal and increments the hung terminal count.

The abbreviations used in the SYSTAT report are defined in Table 4-3.

Table 4-3

SYSTAT Abbreviations

Abbreviation	Meaning
DET	job is detached from all terminals
,	job is not logged into the system but is running LOGIN, SYSTAT, or TTYSET
SW	job is swapped out (not currently in core)
RN	job is running or waiting to run
RS	job is waiting for residency
HB	job is detached (waiting to be attached to a terminal)
FP	job is waiting for file processing action by the system
SL	job is sleeping (SLEEP statement)
CR	job is waiting for card reader input
MT	job is waiting for magtape I/O
LP	job is waiting to perform line printer output
DT	job is waiting for DECtape I/O
PP	job is waiting to perform output on the high-speed paper tape punch
PR	job is waiting for input from the high-speed paper tape reader
TT	job is waiting to perform output to a terminal
KB	job is waiting for input from a terminal
DF	job is waiting to perform disk I/O
AS	device is explicitly assigned to a job
INIT	device is open on a channel
OPR	job runs under a system operator account
DOS	magtape is assigned with DOS/BATCH labeling format
ANSI	magtape is assigned with ANSI standard labeling format
PK	job is waiting for pseudo keyboard I/O
DX	job is waiting for floppy disk I/O
SELF	job runs under the current user account

4.3.1 SYS as a CCL Command

SYS as a CCL command works similarly to SYS typed at a logged out terminal. The CCL command, however, can contain a file specification. The following commands show the proper procedure.

```
SYS B

READY
```

SYSTAT creates file B and writes to it a complete system status report. The file resides under the current user account in the public structure.

```
SYS /B
```

SYSTAT prints a busy device status report at the terminal. To create a status report in a file, type the file specification with an option as follows.

```
SYS  B/B

READY
```

SYSTAT creates a busy device status report in file B.

4.4 PIP PROGRAM

The PIP (Peripheral Interchange Program) system program performs disk and peripheral device transfers as well as several other file utility functions. Two forms of the PIP program are available;[1] the only difference being that one runs in less than 8K words of memory and is generally less powerful than the larger version which requires 16 words to run. Refer to Section 4.18 for a description of the larger version of PIP. (RSTS-11 PIP commands, wherever possible, have been made compatible with DOS/BATCH-11 PIP commands.)

PIP can be called by users logged into the system as follows:

```
RUN $PIP
```

PIP responds by identifying itself and printing a pound sign (#) to indicate that it is able to accept input commands:[2]

```
PIP - RSTS V05B-24 SYSTEM #880
#
```

In order to return to the RSTS-11 Monitor, type CTRL/C or CTRL/Z. The system responds by printing READY. For example:

```
#^Z
READY
```

A CTRL/Z is equivalent to an End-of-File on the user terminal and causes an orderly exit from PIP. Typing CTRL/C causes an immediate exit from PIP and possibly leaves the user's files or directory in a disorderly state.

4.4.1 PIP Command Line Specifications

Spaces and tabs within a PIP command line are ignored. PIP commands must be typed on a single line and be no more than 80 characters long.

Output file specifications are of the form:

```
dev: [proj,prog] name.ext <prot>
```

The elements of the output file specification are described in Table 4-4. Input file specifications are of the form:

```
dev: [proj,prog] name.ext
```

Elements of the input file specification are described in Table 4-5.

---

[1] On systems prior to Version 5C (RSTS/E), two different forms of PIP were distributed: a Record I/O form and a non-Record I/O form. The non-Record I/O PIP performed only formatted ASCII transfers.

[2] On Version 4A-12 (RSTS) systems, PIP prints the asterisk (*) character instead of the pound sign.

Table 4-4

Output File Specification Elements

Element	Description	Default
dev:	device specification	system disk (DF:, DK:, or DP:)
[proj,prog]	account specification, project-programmer number	current user account number
name	filename specification[1]	none
.ext	filename extension[1]	none
<prot>	protection code	<60>, equivalent to read and write protect against everyone but the owner
no specification		KB:

[1]Output-only devices (non-file-structured devices) such as KB:, LP: and PP: ignore the filename and extension specifications.

Table 4-5

Input File Specification Elements

Element	Description	Default
dev:	device specification	system disk (DF:, DK:, or DP:)
[proj,prog]	account specification, project-programmer number	current user account number
name	filename specification[1]	none
.ext	filename extension[1]	.BAS (BASIC-PLUS source program)
no specification		last file specification is used again. Initial default is DF:

[1]Input-only devices (non-file-structured devices) such as KB:, PR: and CR: ignore the filename and extension specifications.

With PIP there is at most one output file, but there may be any number of input files. Where more than one input file is specified, all filenames which are not preceded by a device specification are assumed to be on the system disk. A null file specification duplicates the immediately preceding file specification in all details.

PIP options are specified in the form:

/option:argument

Options are always begun with a slash and terminated with a comma (,), left angle bracket (<), another slash (/), or a line terminating character (RETURN or ESCAPE). The option argument is a function of the individual operation to be performed. The default option is a formatted ASCII file transfer.

4.4.2 File Transfers Including Merge Operations

File transfer and/or file merge operations take the following command format:

#output file<input file(s) /option

Only one output file can be specified. If one input file is specified, a copy of that file is transferred to the output file specification (the original input file remains untouched). If more than one input file is specified, copies of the file are merged into a single output file (the original input files remaining untouched). The options available on a file transfer and/or merge operation are described in Table 4-6.

As an example:

```
PIP - RSTS V05B-24 SYSTEM #880
#DT1:FILET.BAS<FILE1,FILE2,PR:,DT0:AA.BAS
```

The above command string takes the files FILE1 and FILE2 from the system disk, a single file from the high-speed paper tape reader and file AA.BAS from the tape reel mounted on DECtape unit Ø and creates the single file FILET.BAS on the tape reel mounted on DECtape unit 1.

Since a PIP command must be typed on a single line, the number of files which can be merged into a single file is limited only by a command string length of 8Ø characters.

4.4.3 Change Filename or Protection Code

PIP can be used to change a filename specification without transferring any data. The general format for such a command is as follows:

#new file specification = old file specification/RE

To change the filename, extension and/or protection code of a stored file, type the new file specification (including the device specification and, optionally the protection code), followed

Table 4-6

File Transfer and Merge Options

Option	Function
no option	ASCII file transfer is performed.
/FA	Formatted ASCII transfer is performed (Nulls, parity bits[1], and RUBOUT's are ignored).
/BL	Block mode transfer is performed using the default block sizes.[2]
/BL:n	Block mode transfer is performed using a block which is n bytes long.[2]
/CO	Contiguous mode transfer is performed with null fill characters inserted into any partial buffer remaining.[2] Must be used when transferring a .BAC file, a virtual core array, or a file created by Record I/O, from disk to DECtape.
/CO:T	Contiguous mode transfer is performed with any partial buffer remaining being truncated.[2] Must be used when transferring a .BAC file, a virtual core array, or a file created by RECORD I/O from DECtape to disk.
/CL:n	Set cluster size to n.
/GO	Ignore 'USER DATA ERROR ON DEVICE' errors.
/HE	Appends '$PIP.TXT' to command line to have PIP output the helping text explaining PIP commands and options.
/UP	Update file transfer (equivalent to OPEN AS FILE used for output files).
/RW:NO	Disable rewinding of magtape before and after a file transfer.

[1]Parity bits are associated with some ASCII codes, making each ASCII character 8 bits long rather than 7 bits. The DOS/BATCH Monitor system uses even parity ASCII. Even parity implies that the number of bits set within the 8 bit field is an even number.

[2]Available only with the Record I/O version PIP program.

by an equal sign (=), followed by the current file specifications followed by /RE. The current protection code need not be indicated. For example:

```
#FILE.EXT<40>=ABC.DAT/RE
```

In this example, the filename, extension, and protection code are changed.

```
#DT1:MAT.BAC=DT1:ORIG.BAC/RE
```

In this example only the filename is changed. The protection code for the file MAT.BAC remains the same as it was for ORIG.BAC.

In the case where only the protection code is to be changed, a new filename specification need not be typed. A shorter form is:

```
#=old file specification<prot>/RE
```

This command string format indicates that the file protection is to be updated. For example:

```
#=MAG.BAS <48>/RE
```

Notice that the device specification on both sides of the equal sign must be the same. If it is desired to move the file from one device to another, the file merge technique (see section 4.4.2) is used. Note also that PIP assumes a default filename extension of .BAS on input files, but an extension must be specified on output files or none is appended.

4.4.4 File Deletions

To remove a file from the system, type the file specification followed by the /DE (delete) option. For example:

```
#TRY2.BAC/DE
```

This command string causes the file TRY2.BAC to be removed from the system disk if found under the current user's account number. No default assumptions are made as to the filename extension. An extension must be specified if the file was stored with an extension.

The user cannot delete a file under another account or a file which is write-protected against him. If an attempt is made to delete a non-existent file from a device, the error message CAN'T FIND FILE OR ACCOUNT is given.

More than one file can be deleted by specifying several file specifications, separated by commas. For example:

```
*FILE1.BAC,FILE2.BAC,DT1:FILE1.BAC/DE
```

deletes FILE1.BAC from both the system disk and DECtape unit 1 and deletes FILE2.BAC from the system disk. The number of files which can be deleted in one PIP command string is limited only by the maximum length of the command line.

4.4.5 Zero Device Directory

Zeroing a device directory removes all files stored under one account number on the given device. In order to perform this operation, log in to the system under that account number (with the correct password) and run PIP, giving the device specification followed by the /ZE (zero) option:

```
#/ZE
REALLY ZERO [120,80] SY:?
```

If the user does not type a device specification, PIP assumes the system disk and prints the REALLY ZERO account question. To cancel the operation, type the RETURN key. To remove all files from the current account on the system disk, type Y or any string beginning with Y. Note that only the current account can be zeroed. The user must not specify an account number. The following example shows the command to zero a DECtape.

```
#DT1:/ZE
REALLY ZERO DT1:?
```

PIP prints the REALLY ZERO device question since no separation of files by account exists for DECtape. Hence, any user can remove all files from a DECtape reel by the zero action.

If the user is concerned with fast access to files stored under his account, he can restructure his account periodically by use of the /ZE option. The RSTS-11 file allocation mechanism apportions storage space for files on the disk on an as-needed basis. After a user performs numerous file deletion, creation, and extension operations, the internal linkages in his account on a disk point forward and backward in an unordered fashion. To regain an optimum, ordered structure for fast access to files under his account, the user first transfers those files he wishes retained to an external medium. He then uses the /ZE option to remove all files stored under his account from the disk. He then transfers the files to be accessed back to his account. As a result of these actions, the newly assigned linkages (storage and accessing structures) are ordered in an optimum manner for fast access.

4.4.6 List Device Directory

The user can request a listing of all files under his account number, all files with a given name or extension under his account number, or a particular file under his account number on the system disk or any one or more devices. The format of the directory listing command is as follows:

#output<input file(s) /option

An output file specification can be supplied where an output device other than the user terminal is desired. The more usual form of the command is:

#input file(s) / option

in which case the directory listing is sent to the terminal issuing the command. Unless another device is specified, only a directory of the system disk is printed. For example:

```
RUN $PIP
PIP - RSTS V05B-24 SYSTEM #880
#,DT0:,DT1:/DI
```

causes a full directory listing of files for the current user on the system disk (the blank device specification, indicated by the comma with no preceding characters indicates the system disk), and on DECtape units Ø and 1.

The options available with device listings are described in Table 4-7. The input file specifications are described in Table 4-8.

Table 4-7

PIP Directory Listing Options

Option	Function
/BR	BRief directory listing is printed; includes only filenames and extensions with four file specifications on each printed line.
/DI	Normal DIrectory listing is printed; includes filename, extension, length, protection code, and creation date.
/DI:S	Full (Slow) directory listing is printed; includes: a. for disk devices: filename, extension, length, protection code, creation date and time. b. for magtape or DECtape: filename, extension, length, protection code, creation date.

Table 4-8

Input File Specifications

Input File	Directory Printed Includes
none	all files on the system disk under the current user account number.
dev:	all files under the current user account number on the device specified.
dev:*.*	same as the above.
dev:filename.*	all files having the filename specified, under the current user account number, on the device specified.
dev:name.ext	only the file specified, under the current user account number, on the device specified.
Where no device is specified, the default device is the system disk; no message is printed if the particular input file specification(s) cannot be found; where several input file specifications are given, they are separated by commas. Account numbers are only significant for disk files and magtape files.	

4.4.7 Guidelines for Transfer Operations and DECtape Usage

The PIP system program performs transfer operations in any one of several ways depending upon options the user specifies. The options are listed in Table 4-6.

If the user specifies no option in a transfer request, the system checks each byte of data for a CTRL/Z combination CHR$(26). (In Version 4 (RSTS) systems, the CTRL/Z character is defined as the ASCII file terminating character. For compatibility with Version 5 (RSTS/E) systems, it indicates the end of data in the last block of an ASCII file.) PIP transfers the data block by block. The last block transferred by PIP is either the last block in the file or the block containing a CTRL/Z combination.

The ASCII type of transfer is convenient for moving BASIC-PLUS source files from one device to another. For example,

```
#DT1:FILNAM.BAS<FILNAM.BAS
#
```

PIP transfers the source code from the system disk to DECtape and terminates the transfer when the last block in the file is encountered. Any extraneous data following the END statement in the last block of the file causes no harm since the Run Time System recognizes the END statement as the end of file.

Under certain circumstances, PIP can be forced to discard unwanted NUL and RUBOUT characters. The user can specify the /FA option in the transfer request when he creates an ASCII text file from the keyboard or prints an ASCII text file at the keyboard. For example,

```
#KB:<FILNAM.TXT/FA
        (PIP prints the text.)
#
```

PIP terminates the transfer of the disk file upon encountering the CTRL/Z character. Any NUL or non-printing characters are discarded.

The ASCII transfer is not suitable for files containing 8-bit binary data. The bytes in a virtual core array or record I/O file can take any pattern of 8 bits, one of which can be that of the CTRL/Z character. If the user transfers such a binary file with the ASCII type of transfer request, the unpredictable occurrence of the CTRL/Z character pattern causes PIP to terminate the transfer and possibly loses data. To transfer such binary files between like devices, use the /BL option. For example,

```
#DT1:FILNAM.DAT<DTØ:FILNAM.DAT/BL
#DKØ:FILNAM.DAT<DK1:FILNAM.DAT/BL
#
```

PIP transfers the data between the devices strictly block by block and does not check any characters. The default block size of the device is used.

Since both magtape and disk have the same default block size (512 bytes), the user can also specify the /BL option in transfers between magtape and disk. For example,

```
#MTØ:FILNAM.DAT<DK1:FILNAM.DAT/BL
#
```

PIP terminates the transfer when the system signals the end of file on the disk. The file on magtape contains 512 bytes per block.

Because the block sizes of DECtape (510 bytes) and disk (512 bytes) are different, PIP cannot properly handle block by block transfers of binary files between these two devices. If a DECtape block is transferred to disk, two extra bytes are generated on the disk; if a disk block is transferred to DECtape, two bytes are lost. Thus, to transfer binary files from disk to DECtape, the user must specify a special option - the /CO option. For example,

```
#DTØ:FILNAM.DAT<DKØ:FILNAM.DAT/CO
#
```

As a result of the user specifying the /CO option, PIP transfers the file as a stream of contiguous bytes in 510 byte portions. In the last block on DECtape, the system fills any unused locations with NUL characters. In this manner PIP preserves the two extra bytes in the disk block. To transfer a binary file on DECtape to the disk, the user must specify the /CO:T option. For example,

```
#FILNAM.DAT<DTØ:FILNAM.DAT/CO:T
#
```

As a result of the user specifying the /CO:T option PIP transfers the file as a stream of contiguous bytes in 512 byte portions and truncates null characters in the last buffer. This action prevents the length of the file on disk from increasing when no new data is added. (A 10 block file on disk, for example, requires 10 blocks plus 20 bytes - a total of 11 blocks - on DECtape.)

Transferring a compiled file requires a special type of transfer. A compiled file is a unique form of a binary file. (Note that PIP allows only privileged users to transfer compiled files.) When transferring a compiled file, the user must specify one of the options described above and required for binary files. Moreover, if the output device is disk, the user can specify the /CL:4 option. This procedure forces PIP to create the output file in clusters of four contiguous blocks and is helpful since compiled files are always in 1K word clusters on the disk.

The user can transfer a file from a device whose block size is other than 512 bytes by specifying the /BL:n option. For example, to recall a file on magtape created by the BACKUP program with a block size of 2048 bytes, type a command similar to the following.

```
#FILNAM.DAT<MT1:FILNAM.DAT/BL:2Ø48
#
```

PIP reads 2Ø48 byte blocks from the magtape file and creates a disk file using a RECORD SIZE value of 2Ø48.

The /BL:n option can also optimize transfers of large files from disk to disk or from disk to magtape. The advantage is gained by specifying a block size larger than 512. For example, if a large data file resides on a disk, the user can specify the /BL:2Ø48 option in the transfer request. PIP subsequently reads four blocks at a time from the input file rather than executing four separate read operations and writes the output file using a RECORD SIZE value of 2Ø48.

4.4.8 PIP as a CCL Command[1]

The standard CCL table supplied in the RSTS/E generation kit includes PIP as a CCL command. If PIP is included in the CCL for the system, the user may invoke the PIP program simply by typing:

```
PIP
```

This results in exactly the same response as:

```
RUN $PIP
```

Also, typing

```
PIP <string>
```

causes PIP to run, process <string> as a standard PIP command, and return to the READY state. For example, if the following is typed in response to monitor's READY message:

```
PIP LP: <FILE
```

a copy of FILE is printed on LPØ:. Note that whatever was previously in core before the PIP command was executed is now destroyed.

[1]This feature is not available prior to Version 5B (RSTS/E) systems.

THIS PAGE PURPOSELY LEFT BLANK.

4.5 TTYSET PROGRAM

The TTYSET system program is used to establish terminal characteristics for the user terminal. TTYSET can be run by any user before or after he is logged into the system. If the terminal is not logged into the system, only one command can be entered to the TTYSET program at a time in the following format:

```
SET xxxx
```

where xxxx represents one of the TTYSET commands shown in Table 4-9 and is entered with the RETURN or ESCAPE key. If the user is logged into the system, he can run TTYSET as follows:

```
RUN $TTYSET
'TTYSET' TERMINAL CHARACTERISTICS PROGRAM
? XXXX
```

where xxxx is again one of the TTYSET commands shown in either Table 4-9A or 4-9B. When xxxx has been entered to the system with the RETURN or ESCAPE key, TTYSET again prints ? to accept additional commands. To return control to the RSTS-11 Monitor, type EXIT, CTRL/C, or CTRL/Z.

Table 4-9A

RSTS/E TTYSET Commands

Command	Function	Standard Condition
WIDTH n	Set the width of the print line for this terminal to n which can be between 1 and 254. As a result, the system automatically generates a CR and LF sequence if n printing characters have been printed and another printing character is to be transmitted.	See Note 1
TAB	Enable hardware tab control. System transmits HT characters without translation.	
NO TAB	Disable hardware tab control. To move to the next tab stop, the system transmits the correct number of space characters instead of transmitting an HT character.	
FORM	Enable hardware form feed and vertical tab control. System transmits FF and VT characters without translation.	
NO FORM	Disable hardware form feed and vertical tab control. System transmits 4 line feed characters in place of a FF or VT character.	
LC OUTPUT	System transmits the lower case characters CHR$(96) through CHR$(123) inclusive to the terminal without modification.	
NO LC OUTPUT	System translates any lower case character to its upper case equivalent before transmitting it to the terminal.	

[1] See Table 4-10 for the individual characteristics set by RSTS/E.

Table 4-9A (Cont.)

RSTS/E TTYSET Commands

Command	Function	Standard Condition
XON	Special terminal hardware allows the computer to interrupt transmission of characters from the terminal by sending the terminal an XOFF character CHR$(19). Similarly, the computer instructs the terminal to resume transmission of characters by sending the terminal an XON character CHR$(17). The terminal hardware must respond to XOFF and XON characters by ceasing and resuming transmission, respectively.	See Note 1
NO XON	Terminal does not have hardware required for XON feature.	
LOCAL ECHO	Terminal, or its acoustic coupler, echo prints characters as they are generated locally. System does not echo characters received from such a terminal.	
FULL DUPLEX	Characters generated are sent only to the computer. System therefore echoes each character received so that it will be printed locally and translates certain characters to perform the proper action. For example, a CR character received is echoed as a CR and LF character sequence.	
SCOPE	Terminal uses a CRT display and the following characteristics: a. Conforms to synchronization standard (can be conditioned by the STALL command), b. System echoes a DEL character (RUBOUT) as backspace, space, and backspace sequence. c. Terminal generates NUL characters as fill for timing the following operations: home, erase to end of screen, erase to end of line, direct cursor addressing, and line feed.	

[1] See Table 4-10 for the individual characteristics set by RSTS/E.

Table 4-9A (Cont.)

RSTS/E TTYSET Commands

Command	Function	Standard Condition
NO SCOPE	Terminal is not a CRT display device. The system echoes a DEL character (RUBOUT) by printing a \ character and the last character typed and removes the last character typed from the terminal input buffer. Subsequent DEL characters cause the next to last characters to be sequentially printed and removed from the terminal input buffer until a character other than DEL is received. As a result, the system echoes another \ character to delimit the erased characters and echoes the correct character.	See Note 1
LC INPUT	Terminal transmits the full ASCII character set and system does the following: a. Recognizes only the ESC character CHR$(27) as an escape character, b. Echoes and uses the } CHR$(125) and ~ CHR$(126) characters without translation, and c. Echoes and uses lower case alphabetic characters without translation.	
NO LC INPUT	System treats ESC, }, and ~ characters (CHR$(27), CHR$(125), and CHR$(126) respectively) as escape characters and translates lower case characters received to their upper case equivalents.	
NO FILL	System does not generate fill characters for any characters transmitted.	
FILL LA30S	Set the fill characteristics for a serial DECwriter (LA30S).	
FILL n	Set the fill factor to n for this terminal where n is between 0 and 6. As a result, the system generates a multiple of the default number of fill characters for each hardware control character it transmits. See Section 4.5.3 for a discussion of generalized fill characters.	

[1] See Table 4-10 for individual characteristics set by RSTS/E.

Table 4-9A (Cont.)

RSTS/E TTYSET Commands

Command	Function	Standard Condition
SPEEDn[1]	Set to n the rate at which the terminal's interface can accept or pass characters. The value for n can be any number defined for the keyboard number in the system file TTYSET.SPD.	See Note 2.
SPLIT SPEED i/o[1]	Set to i the rate at which the terminal's interface passes input to the computer and set to o the rate at which the terminal's interface accepts output from the computer. The values i and o must be defined for the keyboard number in the system library file TTYSET.SPD.	
NO PARITY	System ignores the parity bit on characters it receives and treats the parity bit on characters it transmits to the terminal as if it were a data bit.	
EVEN PARITY	System sends characters to the terminal with the parity bit properly set for even parity but ignores the parity bit on characters received.	
ODD PARITY	System sends characters to the terminal with the parity bit properly set for odd parity but ignores the parity bit on characters received.	
STALL	Terminal obeys synchronization standard. If the terminal sends an XOFF character (equivalent to the CTRL/S combination), the computer interrupts transmission until the terminal sends either an XON character (equivalent to the CTRL/Q combination) or any other character. If the system receives an XON character,	

[1] These commands can be used only on lines whose interfaces can be programmed to handle more than one speed (e.g., DC11, DH11).

[2] See Table 4-10 for individual characteristics set by RSTS/E.

Table 4-9A (Cont.)

RSTS/E TTYSET Commands

Command	Function	Standard Condition
STALL (cont.)	it does not keep that XON character as data. If the system receives another character, it resumes transmission and keeps that character as data. No characters are lost.	
NO STALL	XON and XOFF characters sent by the terminal have no special meaning.	
UP ARROW	System echoes a control and graphic character combination as the ↑ or ^ characters (CHR$(94)) followed by the proper graphic. For example, CTRL/E prints as ↑E or ^E.	
NO UP ARROW	System echoes control and graphic character combination as is.	
DH BURST n	Set to n the number of characters which a DH11 multiplexer assembles to transmit in a single burst. N can be between 1 and 30 and applies only to terminals on DH11 line interfaces.	
ESC SEQ	System treats an ESC character CHR$(27) and the next incoming character as an escape sequence. The system does not echo either character, but translates the CHR$(27) to CHR$(155) and reverses the characters. If the user executes a GET statement, the buffer contains the second character followed by the CHR$(155) character and RECOUNT is 2.	

Table 4-9A (Cont.)

RSTS/E TTYSET Commands

Command	Function	Standard Condition
NO ESC SEQ	System treats an ESC character CHR$(27) as a line terminating character and echoes it as a $ character.	See Note 1.
EXIT	Terminate a program and return control to the monitor.	
HELP	Print a listing of single and macro commands.	

[1] See Table 4-10 for individual characteristics set by RSTS/E.

Table 4-9B

RSTS (Version 4) TTYSET Commands

Command	Function	Standard Condition
EXIT	Exit from the TTYSET program; return control to RSTS-11 Monitor.	
HELP	Print a description of the other TTYSET commands.	
TAB	Enable hardware tab control by the terminal (significant for ASR-35 Teletypes).	
NO TAB	Disable hardware tab control by the terminal (significant for ASR-35 Teletypes).	Set
FORM	Enable hardware form feed control by the terminal (significant for ASR-35 Teletypes).	
NO FORM	Disable hardware form feed control by the terminal (significant for ASR-35 Teletypes).	Set
LC	Enable input of lower case letters (useful with some terminals, ignored by others).	
NO LC	Translate all lower case letters sent by the terminal into upper case letters.	Set
XON	Enable XON/XOFF remote low-speed reader control.	Set
NO XON	Disable XON/XOFF remote low-speed reader control.	
ECHO	Enable terminal echo facility; results in full duplex mode operation.	Set
NO ECHO	Disable terminal echo facility; results in half duplex mode operation.	
SCOPE	Enable terminal CRT cursor controls (only significant when issued from a CRT display terminal).	
NO SCOPE	Disable terminal CRT cursor controls.	Set

Table 4-9B (Cont.)

RSTS (Version 4) TTYSET Commands

Command	Function	Standard Condition
ESC	Treat only ASCII 033 (octal) code as ESCAPE (ESCAPE or ALT MODE key).	
NO ESC	Treat ASCII 033, 175, 176 (octal) as ESCAPE (ESCAPE or ALT MODE key).	Set
WIDTH n	Set the terminal form width to n characters where $1 \leq n \leq 254$ (octal).	WIDTH 72
SPEED n	Set DC11 baud rate to DC11 speed n where $0 \leq n \leq 3$. The actual baud rate corresponding to n depends upon the hardware setting of the particular DC11 line referenced.	SPEED 0
FILL n	Set fill factor of n, which is multiplied by the default fill factor. See Appendix B of the RSTS-11 System Manager's Guide.	
NO FILL	Set fill factor of 0.	Set
FILL LA30	Set special LA30 serial (DECwriter) fill factor.	
KSR33 ASR33 KSR35 ASR35 VT05 VT06 LA30 LA30S	Treat terminal as though it were the indicated device. These commands automatically select, from the preceding commands, those features applicable to the respective terminal.	Set

In addition to the commands described in Table 4-9, TTYSET in RSTS/E allows the user to set all the individual characteristics for a certain type device by executing one command called a macro command. Each macro command assigns predefined characteristics, any of which can be altered in turn by proper use of an individual characteristics command. Table 4-10 describes the default values for each macro command.

Table 4-10

Default Single Characteristic Settings

Individual Characteristic	Macro Command											
	ASR33	KSR33	ASR35	KSR35	VTØ5	VTØ5B	LA3Ø	LA3ØS	2741	VT50	RTØ2	LA36
WIDTH n	72	72	72	72	72	72	80	80	130	80	32	132
TAB/NO TAB	NO TAB	NO TAB	TAB	TAB	TAB	TAB	NO TAB	NO TAB	TAB	TAB	NO TAB	NO TAB
FORM/NO FORM	NO FORM	NO FORM	FORM	FORM	NO FORM	NO FORM	NO FORM	NO FORM	NO FORM	NO FORM	NO FORM	NO FORM
LC OUTPUT/ NO LC OUTPUT	NO LC OUTPUT	NO LC OUTPUT	NO LC OUTPUT	NO LC OUTPUT	LC OUTPUT	LC OUTPUT	NO LC OUTPUT	NO LC OUTPUT	LC OUTPUT	NO LC OUTPUT	NO LC OUTPUT	LC OUTPUT
XON/NO XON	XON	NO XON	XON	NO XON	NO XON	NO XON	NO XON	NO XON	NO XON	NO XON	NO XON	NO XON
LOCAL ECHO FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX	FULL DUPLEX
SCOPE/ NO SCOPE	NO SCOPE	NO SCOPE	NO SCOPE	NO SCOPE	SCOPE	SCOPE	NO SCOPE	NO SCOPE	NO SCOPE	SCOPE	NO SCOPE	NO SCOPE
LC INPUT/ NO LC INPUT	NO LC INPUT	NO LC INPUT	NO LC INPUT	NO LC INPUT	NO LC INPUT	NO LC INPUT	NO LC INPUT	NO LC INPUT	LC INPUT	NO LC INPUT	NO LC INPUT	LC INPUT
FILL n FILL LA30S	Ø	Ø	1	1	Ø	3	Ø	FILL LA3ØS	2	Ø	1	1
SPEED n SPLIT SPEED i/o 2741	11Ø	11Ø	11Ø q	11Ø	3ØØ	15Ø/2400	3ØØ	3ØØ	2741	3ØØ/96ØØ	11Ø	3ØØ
NO PARITY EVEN PARITY ODD PARITY	NO PARITY	NO PARITY	NO PARITY	NO PARITY	NO PARITY	NO PARITY	NO PARITY	NO PARITY	NO PARITY	NO PARITY	EVEN PARITY	NO PARITY
STALL/ NO STALL	NO STALL	NO STALL	NO STALL	NO STALL	STALL	STALL	NO STALL	NO STALL	NO STALL	STALL	NO STALL	NO STALL
UP ARROW NO UP ARROW	UP ARROW	UP ARROW	UP ARROW	UP ARROW	NO UP ARROW	NO UP ARROW	UP ARROW	UP ARROW	NO UP ARROW	NO UP ARROW	NO UP ARROW	UP ARROW
DH BURST h	3Ø	3Ø	3Ø	3Ø	3Ø	3Ø	3Ø	3Ø	3Ø	12	3Ø	3Ø
ESC SEQ/ NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ	NO ESC SEQ

The TTYSET program under RSTS/E checks commands before and during execution and reports errors by printing the messages described in Table 4-11. If any errors involve the terminal speed file, the user should immediately notify the system manager or responsible system programmer.

Table 4-11

RSTS/E TTYSET Error Messages

Message	Meaning
<string> IS AN ILLEGAL KB SPECIFICATION	The keyboard number denoted by <string> is not between Ø and 63 or is not a valid number.
ILLEGAL FILL FACTOR	Fill factor specified is not between Ø and 6.
ILLEGAL WIDTH	Width specified is not between 1 and 254.
ILLEGAL DH BURST	Burst must be between 1 and 3Ø.
COMMAND <string> ILLEGAL	Command indicated by <string> is undefined.
ILLEGAL SPEED	Speed given is not one defined for the device in the $TTYSET.SPD file.
WARNING - ERROR READING $TTYSET.SPD FILE	Program warns user of possible corruption in the terminal speed file and denotes the exact error by printing the COMMAND ERROR message.
COMMAND ERROR - <text>	The RSTS error denoted by <text> was encountered when executing the command.
ERROR - <text>	The RSTS error denoted by <text> was encountered when executing the system function to change terminal status.
WARNING - CANNOT OPEN $TTYSET.SPD FILE	Program could not access the terminal speed file and denotes the exact error by printing the COMMAND ERROR message.

4.5.1 ESCAPE, ALTMODE, and PREFIX Characters

The RSTS-11 system translates certain ASCII characters in a special manner. Some terminals have the outmoded ASCII character keys ALTMODE (125 decimal) and PREFIX (126 decimal). More recently designed terminals incorporate the 1968 ASCII character set and include the following control characters:

ESCAPE = 27 (decimal)
} = 125 (decimal)
~ = 126 (decimal)

The RSTS-11 system interprets CHR$ (27) as a line terminating character and automatically translates any CHR$ (125) and CHR$ (126) characters input from a terminal into 27 (decimal). The system thus treats 125 (decimal) and 126 (decimal) as line terminators.

A user having a terminal with the 1968 ASCII character set can make the system treat 125 (decimal) and 126 (decimal) characters as they are printed rather than as control characters. The TTYSET commands LC INPUT and NO LC INPUT alter internal parameters for a given terminal so that the system treats 125 (decimal) and 126 (decimal) as they are printed or translates them automatically[1].

4.5.2 Lower and Upper Case Characters

Some terminals can send and print both lower case and upper case characters. Such terminals can therefore print the echo returned by the system to give the user an accurate visual representation of the character transmitted.

---

[1] On Version 4 (RSTS) systems, the selectable ALTMODE translation option must be included on the system and the ESC and NO ESC commands must be used to alter the internal parameters.

Terminals such as the VT05 alphanumeric display can send either lower case or upper case characters but can print only upper case characters. Consequently, the echo response of such a terminal to a lower case character is the upper case counterpart. The terminal gives the user no visual indication that the character transmitted was a lower case character.

Terminals such as the ASR-33 and KSR-33 type devices neither send nor receive lower case characters. If such a terminal receives a lower case character, it prints the corresponding upper case character.

Normally, RSTS-11 software translates all lower case characters to their upper case counterparts before processing them. To take advantage of different features of terminal hardware, a RSTS/E user chooses or omits lower case translation depending upon whether the terminal produces lower case output, lower case input, or both.[1] The LC OUTPUT and LC INPUT commands, respectively, omit translation of lower case characters generated as output on and input from a terminal. NO LC OUTPUT causes lower to upper case translation on output to the terminal; NO LC INPUT causes lower to upper case translation on input from the terminal.

4.5.3 Generalized Fill Characters

The RSTS-11 system automatically generates a variable number of NUL characters (CHR$(0%)) as fill characters after processing certain control characters[2]. The generation of the meaningless NUL characters allows the terminal sufficient time to complete the physical action initiated by the control character and permits the terminal to synchronize itself properly for printing the next meaningful character. The control characters and the related default number of fill characters generated are shown in Table 4-12.

---

[1] On Version 4 (RSTS) systems, the selectable lower to upper case translation option must be included on the system to be able to omit automatic translation. In addition, the LC and NO LC commands control automatic translation of all lower case characters, whether generated as input or output.

[2] On Version 4 (RSTS) systems, the terminal fill option must be included on the system for the automatic generation of fill characters to occur.

Table 4-12
Control Characters Requiring Fills

Control Characters	Decimal Value	Default Number of NUL Characters Generated
CR (RETURN)	13	1
LF	10	1
HT (TAB)	9	1
VT	11	4
FF	12	9

Since the default number of fill characters is appropriate only for a terminal operating at 110 baud, the user must alter the number of NUL characters generated as fill for a terminal operating at a speed greater than 110 baud. The TTYSET commands FILL and NO FILL selectively generates multiples of the default number of NUL characters or disables automatic generation of NUL characters. The multiple is called the fill factor. If the user types the FILL 2 command, the system subsequently generates twice the number of default NUL characters - 2 NUL characters after CR, LF, and HT characters; 8 NUL characters after VT characters; and 18 NUL characters after FF characters. To disable the generation of fill characters, the user can type the NO FILL command which effectively sets the fill factor to 0.

4.5.4 XON/XOFF Remote Reader Control

To operate a low speed paper tape reader connected to the RSTS-11 system by either a data set (dial up) or a communications line, two requirements must be fulfilled as follows[1].

a. The terminal must be equipped with the requisite hardware option for XON/XOFF remote reader control

b. The XON/XOFF code must be enabled for the given terminal.

---

[1] On Version 4 (RSTS) systems, the XON/XOFF remote reader control option must be included on the system at system generation time to operate a low-speed paper tape reader as described.

The user can selectively enable and disable remote reader control for a remote terminal by the TTYSET commands XON and NO XON.

For low speed readers connected to the system by a local line interface, none of these requirements is necessary.

4.5.5 Output Parity Bit

DEC-supplied terminals are normally wired to send non-parity characters and to ignore the parity bit on characters received. Consequently, the RSTS-11 software normally ignores the parity bit on characters received from a terminal and omits the parity bit on characters it transmits[1]. Since some terminals not suppled by DEC can receive even or odd parity characters, the software must be conditioned to send parity characters. The TTYSET commands EVEN PARITY, ODD PARITY, or NO PARITY condition the software to send the correct parity. The software ignores parity bits on characters input to the system.

4.5.6 SET as a CCL Command

Under a standard RSTS/E system, the user can execute TTYSET by the correct concise command language (CCL) command. To execute a CCL command, merely include a valid TTYSET command following the proper CCL command. For example,

```
SET ASR33
READY
```

For more information on CCL commands, see the introductory material for Chapter 4.

[1] On Version 4 (RSTS) systems, the no parity option must be included on the system at system generation time if the software is to automatically omit the parity bit on characters it transmits.

THIS PAGE PURPOSELY LEFT BLANK.

4.6 QUOLST PROGRAM

The QUOLST system program allows the user to determine what portion of his disk quota is currently occupied and the number of free blocks remaining on the system disk. QUOLST is called as follows:

```
RUN$  QUOLST
```

Output from QUOLST includes the user account number and information printed under the following headings:

Table 4-13

QUOLST Column Headings

Column Heading	Meaning
STR	STRucture, device being reported.
USED	number of used 256-word blocks under the user account.
FREE	number of free blocks remaining in the user account disk quota.
SYSTEM	number of free blocks remaining to the system on the structure indicated.

Output from QUOLST looks as follows:

```
RUN  $QUOLST

USER:   [1ØØ,1ØØ]
STR     USED     FREE     SYSTEM
DF:     55       145      1234
DK1:    1Ø       1992     1992
```

In this example, the user is logged into the system under account [1ØØ, 1ØØ] and has used 55 blocks on the system disk(s) with a quota of 2ØØ blocks (2ØØ -55=145 free blocks). There are 1234 free blocks on the system disk(s). User [1ØØ, 1ØØ] also has access to private disk DK1: He has used 1Ø blocks on DK1: and has no disk quota on that device; therefore, the free user block count is equal to the free system block count (1992 blocks).

THIS PAGE PURPOSELY LEFT BLANK.

4.7 MONEY PROGRAM

MONEY is the RSTS-11 system accounting program which allows a user to obtain printed data on his own account status. The program is called as follows (only by a user logged into the system):

RUN$ MONEY

The following shows Teletype output resulting when a user logged into the system under account [100,100] runs the program MONEY:[1]

```
RUN $MONEY

 ACCT    PASSWORD     CPU-TIME     KCT'S CONNECT DEVICE   DISK  QUOTA  UFD
100,100                   2.5       124       3      0      0    400   16

READY
```

[1] On Version 4A-12 systems, MONEY prints an ACCESS column which gives two numbers: the number of logins and the number of logouts.

THIS PAGE PURPOSELY LEFT BLANK.

4.8 GRIPE PROGRAM

The GRIPE system program allows a user to communicate comments to the system manager. Comments which the user types while running GRIPE are written to a common file where they are retained for inspection by the system manager.

The user runs GRIPE by typing the following command.

```
RUN$GRIPE
```

GRIPE indicates that it is ready to accept user comments by printing a query line as follows:

```
YES? (END WITH ESCAPE)
```

The user is then allowed to type the text of his comment which is entered into the common file. The user terminates the text of his comment by typing the ESCAPE or ALTMODE key. (Typing the ESCAPE or ALTMODE key is echoed at the terminal by a dollar sign ($) being printed. No carriage return-line feed operation is performed. The program indicates its acceptance of the text and its termination by printing the following lines.

```
THANK YOU
READY
```

THIS PAGE PURPOSELY LEFT BLANK.

4.9 EDIT PROGRAM[1]

The EDIT system program is used to prepare and modify text or program files. It can be run by any user. To run EDIT, the user types:

```
RUN $EDIT
EDIT V05-03

#
```

In response to the number sign (#), the user must specify the input files he wishes to modify and the files to be created as output. The form of this specification is:

```
#OUT1,OUT2<IN1,IN2/B
```

where IN1 and OUT1 are the primary input and output files, respectively, and IN2 and OUT2 are the secondary input and output files. These file names may be any valid RSTS-11 file specifications, including a device, name, extension, project-programmer number, and protection code. The "/B" is included if the user desires to edit a BASIC-PLUS program file containing LINE FEED continuation of lines. (See the BASIC-PLUS Language Manual, Section 2.3.2.) Of these file specifications, only the primary output file, OUT1 must be included. This file alone being specified is used to indicate the creation of a new file.

If IN1 and OUT1, the primary input and output, are specified as the same file, the primary input file will be renamed after the editing is complete to have an extension of BAK, the "backup" file. In this case EDIT will use a temporary name of EDITnn.TMP for the primary output file during editing operations (where "nn" is the job number under which EDIT is being executed). If the secondary output file is a line printer (LPn:), the printer will only be assigned when needed for output. (If the printer is unavailable when needed, the user is given the option of waiting for some time of his choosing or of aborting the output request.)

Once EDIT has been given the file specifications, it will open the necessary files and respond with an asterisk (*), indicating it is ready to accept editing commands from the user. The valid commands are as described in the PDP-11 Edit-11 Text Editor Programmer's Manual, DEC-11-UEDAA, and are summarized in Table 4-14 here.

[1] The EDIT program is not available prior to Version 5B(RSTS/E) systems.

The EDIT-11 command T (trailer) has no meaning to the RSTS-11 EDIT program. All other descriptions of EDIT-11 commands and how they operate are valid for RSTS-11 EDIT as well. If an error occurs in executing a command as typed by the user, the command already executed will be printed, terminated with a question mark (?) at the point at which the error was noted. When editing is complete, EDIT returns to the request for file specifications (#).

4.9.1 EDIT as a CCL Command[1]

EDIT may be run as a CCL command by typing one of the following:

1) EDIT
2) EDIT OUT1,OUT2 < IN1,IN2
3) EDIT FILENAME
4) CREATE FILENAME

The first of these is equivalent to: RUN $EDIT. The header line is printed, and a # character prompts the user to specify his files. The second CCL command, shown above, runs EDIT and also automatically sets up the input and output files as specified. No header line is printed.

The third CCL command is equivalent to the set of commands:

```
RUN $EDIT
FILENAME <FILENAME/B
R
```

except that the header line of EDIT is not printed. EDIT runs, sets up a .BAK file, reads the first buffer, and waits for the user to type editing commands.

The last CCL command shown above sets up the file named FILENAME as an output file. No .BAK file is set up; there is no input file; and no header is printed. If the file existed previously, it is reduced to zero length. As a first command, type I (insert). The CREATE command is equivalent to the instructions:

```
RUN $EDIT
FILENAME
```

[1]This feature is not available prior to Version 5B (RSTS/E) systems.

Table 4-14

Summary of EDIT Program Commands

Command	Format[1]	Result
Read	R	Read from primary input file until form feed is encountered or all internal buffer space is filled.
Edit Read	ER	Read from secondary input file until form feed is encountered or all internal buffer space is filled.
Write	nW	Write n lines into primary output file, starting from the current position Dot.
Edit Write	nEW	Write n lines into secondary output file, starting from the current position Dot.
Next	nN	Write the contents of the Page Buffer into the primary output file, kill the buffer, and read a page of text from the primary input file. Repeat n times. Equivalent to B/W /D R.
Form feed	nF	Insert n form feed characters at Dot.
Beginning	B	Move Dot to the beginning of the Page Buffer.
Advance	nA	Advance Dot n lines. Leaves Dot at beginning of line.
Jump	nJ	Move Dot over n characters.
Delete	nD	Delete n characters from text, starting at Dot.
Kill	nK	Kill n lines of text, starting at Dot.
Mark	M	Mark the current location of Dot.
Save	nS	Save the next n lines in the Save Buffer, starting at Dot.
Unsave	U	Copy the contents of the Save Buffer into Page Buffer at Dot.
List	nL	List n lines on teleprinter, starting at Dot.
Verify	V	Verify the present line via teleprinter.
Get	nG#XXXXX# or nG <CR> XXXX<CR> <LF>	Search for the n^{th} occurrence of XXXXX. Return with Dot following XXXXX.
wHole	nH#XXXXX# or nH <CR> XXXX <CR> <LF>	Search for the n^{th} occurrence of XXXXX. If found on this page, return with Dot following XXXXX. If not found, execute an N command and continue search.

[1] In the table, # represents any ASCII character. <CR> represents a return character, and <LF> represents a line feed character.

Table 4-14 (Cont.)

Summary of EDIT Program Commands

Command	Format[1]	Result
Edit wHole	nEH#XXXXX# or nEH <CR> XXXX< CR > < LF>	Perform a wHole search for the n^{th} occurrence of XXXXX, using the secondary input and primary output files.
Position	nP#XXXXX# or nP< CR > XXXX< CR > < LF>	Perform a Next command, then search for the n^{th} occurrence of XXXXX. If found, return with Dot following XXXXX. If not found, clear the buffer, read another page, and continue search.
Edit Position	nEP#XXXXX# or nEP<CR > XXXX <CR > < LF >	Perform a Position search using secondary input rather than primary input file.
Insert	I#XXXXX# or I <CR> XXXX< CR > < LF>	Insert the text XXXXX at Dot. Move Dot to follow XXXXX.
Change	nC#XXXXX# or nC <CR > XXXX <CR > < LF >	Change n characters starting at Dot to XXXXX. Equivalent to Insert followed by n Delete.
eXchange	nX#XXXXX# or nX <CR > XXXX <CR > < LF >	eXchange n lines starting at Dot for XXXXX. Equivalent to Insert followed by n Kill.
Execute Macro	nEM	Execute the first line of the Save Buffer as a command string n times.
Exit	EX	Perform consecutive Next commands until end of primary input file is reached. Close all files, and return to file specification request.
Edit Open	EO	Move to beginning of the secondary input file.
End File	EF	Close the primary output file to any further output and close the primary input file.

[1] In the table, # represents any ASCII character. <CR> represents a return character, and <LF> represents a line feed character.

4.10 BACKUP PROGRAM[1]

The BACKUP system program is employed on-line to preserve and recall files stored under one user account or under several user accounts. It provides a means of transferring multiple files from a private disk or from the public disk structure to a private disk, to a magtape, or to a DECtape. BACKUP also provides the means of later transferring those files back to the original device. To run, BACKUP requires that the Record I/O software option be included in the system configuration. The BACKUP program and its related program BACKDK are stored in their compiled forms in the system library with protection code <104>.

BACKUP is the command scanner and interpreter for the BACKDK program. BACKDK actually performs the transfer operations. A command typed to BACKUP is parsed and passed to error checking routines. Most error reporting and checking for file and account protection violations are done by BACKUP. When no errors are detected in the command, BACKUP writes the valid, parsed command to the core common area and chains to the BACKDK program in the system library.

BACKDK runs and transfers the designated files according to options given in the command. File protection code and cluster size information is preserved for all files except for those transferred to and from DECtape. Before transferring a file to disk or to DECtape, BACKDK performs an OPEN FOR INPUT operation to determine if a file of the same name and extension currently exists. Unless the user specifies otherwise in the BACKUP command, BACKDK does not transfer the file if it currently exists on the output disk or DECtape device. BACKDK transfers files to magtape without checking for the current existence of the files.

Any protection violation errors encountered when transferring a file causes BACKDK to print the INTERLOCK error message, to skip the file, and to proceed without interruption. Files open in UPDATE mode while BACKDK is running are not transferred. When an end of device occurs during transfers to or from magtape, BACKDK allows the user to continue with another reel. If a fatal error occurs during a transfer, BACKDK prints the related error message and chains to the BACKUP program.

[1] This feature is not available prior to Version 5B (RSTS/E) systems and is not available on Version 4B (RSTS) systems.

Both privileged and non-privileged users can run BACKUP. BACKUP interprets all requests by a non-privileged user as applying to the account under which the program is currently running. Only those files stored in his account are affected. In addition, the date of creation and date of last access information for files transferred by a non-privileged user are always reset to the current system date. (Technically, BACKUP always enables the /B option for a non-privileged user.) Also, a non-privileged user can not transfer a compiled (run-only) file. If a non-privileged user attempts to transfer a compiled file, BACKDK prints the PROTECTION VIOLATION error message and proceeds to the next file.

A privileged user can specify a project-programmer field in a command. This facility allows him to transfer files stored under accounts and to specify multiple accounts in a single transfer request. Date of creation and date of last access information is preserved for files transferred by a privileged user unless he specifies the /B option. BACKUP provides several options which can be executed only by a privileged user. These options are described in Section 4.10.7.

BACKUP uses a command string similar to the PIP system program and can be considered a superset of Record I/O PIP. BACKUP performs no format checking and, consequently, generates no error if, for example, a request is made to transfer a binary file to a character oriented device such as the line printer.

4.10.1 Running and Terminating BACKUP

The BACKUP system program is run from a terminal logged into the system by typing the following command.

```
RUN $BACKUP
```

When BACKUP runs, its prints a header line containing the program name and version number, followed by the pound sign (#) character as shown below.

```
FILE BACKUP  VØ5-23
#
```

The user can type a command string to BACKUP any time BACKUP prints the # character. Typing a command string beginning with H causes a help message to be printed at the terminal.

The user terminates BACKUP by typing either the CTRL/C or CTRL/Z combination in response to the # character printed at the terminal.

```
#↑Z
READY
```

The system indicates the termination of BACKUP by printing the READY message.

4.10.2 BACKUP Command Specification

The BACKUP system program accepts a command string similar to that accepted by the PIP system program. Commands must be typed on a single line and contain no more than 80 characters. Spaces, tabs, and commas are ignored by the program but included in the 80-character limit. A command to transfer files has the following general form:

```
#output specification<input specification/options
```

The file or files designated by the input specification are transferred to the output device. The output specification consists of either a valid device and unit designator or no specification. Giving an account number in the output specification is not valid but generates no error. Omitting the output device causes BACKUP to use the public disk structure.

Valid device designators are the same as those defined for the input specification in Table 4-15. The additional output device is allowed: line printer units Ø through 7 (LPØ: through LP7:). One output specification is allowed per command. The left arrow character (<) separates the output specification from the input specification.

The input specification is of the following form:

```
dev:<dev:[proj,prog]filename.ext
```

The elements of the input specification are described in Table 4-15. One input specification or no input specification is allowed for each BACKUP command. No input specification indicates all files on the public disk structure for the current user account. The [proj, prog] designation is valid for privileged users only. For non-privileged users, BACKUP automatically changes any [proj,prog] designation given in an input specification to the project-programmer number of the current account.

Table 4-15

BACKUP Input File Specification Elements

Element	Format	Meaning	Examples
Project-programmer field	[proj,prog] [* ,prog] [proj,*] [*,*] null	User account number. The appearance of an asterisk means that files for either all project numbers or all programmer numbers or both are designated for the transfer. Never can be [Ø,1]. If null, files for only the current account are transferred.	[2ØØ,2ØØ] [*,2ØØ] [2ØØ,*] [*,*]
Device designator	DKØ: to DK7: DPØ: to DP7: DTØ: to DT7: MTØ: to MT7: null	RKØ5 or RKØ3 DECpack units Ø through 7 RPØ3 disk pack units Ø through 7 DECtape units Ø through 7 Magtape units Ø through 7 Public disk structure	DK2: DP1: DT1: MT1:
Filename and extension	filename.ext filename.* *.ext *.* null	An asterisk in the filename specification means all filenames; in the extension specification, all extensions (including filename with no extension). The conventional explicit filename and extension specification is also recognized. Null is equivalent to specify all files (*,*).	FILE.BAS FILE.* *.BAS *.*
	file??.ext file.e?? file??.e??	The question mark character in any position of either a filename or extension specification or of both means that any alphanumeric character can be substituted for the ? to designate a file to transfer. In this manner, files with similar filenames and extensions can be easily designated for backup or recall.	FILE??.BAS FILE.B?? FILE??.B ??FILE.?A?
	file??.* *.ex?	The asterisk and question mark can be used together.	FILE??.* *.BA?
Protection Code	N/A	Protection codes are transferred in all transfers except those involving DECtape. Files cannot be transferred conditionally based on protection code.	

One or more BACKUP options can be specified following the input specification or can appear as a command by itself. An option is specified in the following general format:

/option:argument:argument

An option always begins with a slash character (/) and terminates with another slash (the start of another option) or terminates with a line terminating character (RETURN). The options are listed and described in Table 4-16 (next page).

4.10.3 Magtape Characteristics

Magtape devices are non-directory oriented. The system writes only contiguous files to magtape and writes new files following those files currently on the reel. Therefore, the BACKUP program cannot replace an older version of a file on magtape with a newer version and cannot selectively delete files currently written on magtape. As a result, the /S and /D:dd-mm-yy options have no effect when the input device is magtape

Immediately following the last file written on a magtape are three end of file (EOF) marks. When the user specifies magtape as the output device, BACKUP reads the magtape until it detects the standard three consecutive EOF marks. The program then backspaces and writes the first file over the last two end of file marks. The program writes files until the end of tape is sensed, after which it backspaces to the last EOF mark, writes 3 EOF marks to signify the end of tape, and prints a message allowing the user to continue output to another reel on another unit or on the same unit.

Since files with the same filename specifications can exist on a magtape reel, the user must exercise care when he transfers files to magtape. It is suggested that the user zero the magtape before transferring files to it. To do this, he can specify the /Z option in the BACKUP command. When the user specifies the zero action, BACKUP writes 3 consecutive EOF marks after the beginning of the tape and begins the transfer at the beginning of the magtape.

BACKUP maintains additional information in the standard 7-word file label record on magtape. Byte 9 contains the cluster size minus 1 of the file and bytes 12 and 13 contain the date of last access for the file. These bytes are not used by the RSTS-11 magtape file system. However, DOS magtape expects these bytes to be zero, so a BACKUP magtape may not be readable under DOS.

Table 4-16

BACKUP Options for General Usage

Option	Meaning
/C:dd-mmm-yy	Transfer only those files having a creation date greater than or equal to the date specified by dd-mmm-yy. For example, /C:12-JAN-74 transfers only those files with a creation date on or after January 12, 1974.
/A /A/L:device /A/L:filename.ext	Prints a directory listing at the user's terminal. Prints a directory listing at the device. Prints a directory listing in the file specified in the /L option. BACKUP performs no transfer.
/D:dd-mmm-yy	Delete those files on the input device after the transfer if their dates of last access are less than the date specified by dd-mmm-yy. For example, D:12-JAN-74 deletes those files whose date of last access in January 11, 1974 or before.
/I:A /I:F /I	Selectively transfers files by account number (I:A) for privileged users or by filename (I:F) for general users. BACKDK prints each account and/or each filename eligible for selection and asks the user if he wishes to perform a transfer. Specifying /I indicates selection by account and filename (I:A:F). Can be used with the /V option as well as with transfer of multiple files or accounts.
/L /L:device /L:filename.ext	Prints listing information at the user's terminal. Prints listing information at the device indicated. Prints listing information in the file specified (L:filename.ext). A disk device or protection code can be included in the file specification (L:DK1:FILE.LST<40>).
/R	Used only for recall of files from DECtape. Prevents lengths of files created by BACKUP from inflating unnecessarily.
/S	Supersedes currently existing files on DECtape or disk with files of same name on the input. BACKUP does not ordinarily supersede existing files.
/V	Reads each block of each file specified on the device indicated. No transfer is performed. Verifies accessibility of files by determining whether they contain bad blocks.
/Z	Equivalent to the /ZE option of the PIP system program. User must type Y or any string beginning with Y in response to the REALLY ZERO query to perform the zero action.

4.10.4 Disk Characteristics

The user can employ a private disk pack to preserve files of the public disk structure. The DECpack or disk pack must be logically mounted and in the UNLOCK state before BACKDK can write files to or read files from the pack. Each account on the public structure from which or to which BACKDK transfers files must exist on the private pack. If an account does not exist on the private disk pack, the files of that account are not transferred. BACKDK reports the RSTS-11 error number 5 and continues processing any further accounts.

Since disk is a directory oriented device, the BACKDK program must search an account directory for the existence of each file transferred. BACKDK usually does not transfer a file from the input device if a file of the same filename and extension currently exists on the output disk under the same account. BACKDK simply omits the transfer, flags the file with the ALREADY EXISTS message, and proceeds to the next file. To have BACKDK supersede currently existing files on the output device, the user must specify the /S option in the command to BACKUP.

BACKDK can not transfer files which are open in UPDATE mode.

4.10.5 DECtape Characteristics

DECtape is a directory oriented device as is disk. A directory entry on DECtape maintains filename, filename extension, length, and date of creation information. Date of last access, protection code, and project-programmer number information are not used when reading and writing DECtape under RSTS-11. Files from more than one account should never be transferred to the same DECtape reel if user account and protection code information are important. When transferring files from DECtape, BACKDK uses the account on the output device corresponding to the account under which the program is running, sets the protection code of each file to 60, and sets the date of creation, time of creation, and date of last access to the current system date and time. If a privileged user specifies a single account in the input specification project-programmer field when he transfers files from DECtape, the BACKDK program writes the files to that account on the output device.

All transfers to and from DECtape are performed on a packed block-by-packed block basis. Consequently, a transfer made from a disk to DECtape causes more blocks to be generated on the DECtape than originally existed on disk. Because of the smaller block size on DECtape (510 bytes per block), a file on disk requires a larger number of blocks when transferred to DECtape. For example, ten blocks on disk require ten blocks plus 20 bytes or eleven blocks on DECtape.

Because of the smaller block size of DECtape, a disk file created by BACKDK on DECtape grows in length when transferred back to disk. Repeated transfers of disk files to DECtape and back causes the size of the file to increase although no additional information is added. To prevent unwarranted inflation in the size of a file created by BACKUP, use the /R option in the BACKUP command when recalling that file from DECtape.

Do not use the /R option when recalling a file from DECtape if that file was not created by BACKUP. Information at the end of the file on DECtape can be lost.

If BACKDK is transferring files to DECtape and the NO ROOM FOR USER ON DEVICE error occurs, the program prints an error message followed by an information message concerning the number of blocks and files transferred. BACKDK then chains to the BACKUP program which prints the pound sign character. An incomplete file and the directory entry for the incomplete file may exist on the DECtape.

BACKDK omits transferring certain files to and from DECtape if those files of the same filename and extension currently exist on the DECtape. To supersede existing copies of files, the user must specify the /S option in the BACKUP command.

Because BACKDK must search the directory on DECtape for each file transferred to or from DECtape, operations using DECtape tend to be slow. In addition, account information is not recognized on DECtape. For these reasons, DECtape is more beneficial to the general user who wants to preserve a relatively small number of files from one account. The maximum usable storage capacity on DECtape is 561 blocks, each 510 bytes in length. To preserve a large number of files from many accounts, magtape or private DECpack or disk pack is more advantageous.

4.10.6 General Usage of BACKUP Commands

This section describes the general usage of BACKUP commands and options for both the privileged and the non-privileged user.

To transfer all files from the current account in the public disk structure to a DECtape, the user can type a command similar to the one shown in the following sample dialog.

```
#DT1:<*.*/Z/L:BACKUP.LST
REALLY ZERO DT1: ? Y

TRANSFER COMPLETE  365 BLOCKS TRANSFERRED IN 10 FILES
#
```

The /Z option causes BACKDK to print the REALLY ZERO query. To zero the device, the user must type Y or any string beginning with Y. If the user decides not to zero the device, he can type the RETURN key only or any string not beginning with Y. BACKDK prints listing information in the file BACKUP.LST on the system disk. Upon completion of the transfer, BACKDK prints the TRANSFER COMPLETE message and chains to BACKUP which prints the # character.

To recall files created by BACKDK on DECtape, the user can type the following command.

```
#SY:<DT1:*.*/R/L:RECALL.LST
TRANSFER COMPLETE  365 BLOCKS TRANSFERRED IN 10 FILES
#
```

The /R option prevents inflating the size of files created by BACKDK. Listing information is placed in the file RECALL.LST on the system disk. As a result of the command, all files on the DECtape are transferred to the current user's account on the system disk. Those files currently on the disk are not superseded.

To transfer certain files selectively to a magtape, use a command similar to the one shown in the following sample dialog.

```
#MT1:<[*,*]PROG??.*/Z/I/L:BACKUP.LST
REALLY ZERO MT1: ? Y
[111,26]? Y
PROGØ1.BAS? Y
PROGAA.BAS?

#
```

The /Z option causes BACKDK to print the REALLY ZERO query. To zero the magtape, the user types Y or any string beginning with Y. Typing the RETURN key only or any string not beginning with Y causes BACKDK to search for the end of the last file on tape before beginning the transfer. The files on tape are not overwritten.

The /I option causes BACKDK to print, as a query, the project-programmer number and file name of each account and file eligible for transfer. To include the item in the transfer, type Y or any string beginning with Y in response to the query. Typing the RETURN key only or any string not beginning with Y causes BACKDK to omit the item from the transfer. (For a non-privileged user, BACKDK prints the current account number as the only eligible project-programmer number.) Listing information is written to the file BACKUP.LST.

If the end of magtape is reached, BACKDK prints the message NEXT MAGTAPE UNIT #, after which the user must type a number between Ø and 7 inclusive. This number designates the magtape unit on which the continuation reel is mounted. Subsequently, BACKDK continues transferring files to that reel. (If the user originally specifies the /Z option, BACKDK prints the REALLY ZERO message for each continuation reel.)

The user can transfer files to a private disk by typing a command similar to the following command.

```
#DK1:<*.BAS/S
TRANSFER COMPLETE  112 BLOCKS TRANSFERRED IN 4 FILES
#
```

The /S option ensures that BACKDK supersedes any files currently in the user's account on the disk with files of the same filename and extension from the system disk. The command in the sample dialog causes BACKDK to transfer all BASIC-PLUS source files to the private disk pack mounted in unit 1. Upon completion of the transfer, BACKDK prints the information message telling the number of blocks and files involved in the transfer and prints, on a subsequent line, the pound sign character.

To print the directory of a device, the user can specify the /A option as in the following command.

```
#DT0:/A

  111 , 26 = PPN BEING LISTED AT 02:21 PM  27-FEB-74

DT0:SORT  .HLP    7       27-FEB-74 01-AUG-73  02:21 PM 0
DT0:SORT  .BAS    26      27-FEB-74 01-AUG-73  02:22 PM 0
DT0:XQWIK .BAS    23      27-FEB-74 01-AUG-73  02:22 PM 0
DT0:UQWIK .BAS    17      27-FEB-74 01-AUG-73  02:22 PM 0
DT0:SQWIK .BAS    14      27-FEB-74 01-AUG-73  02:22 PM 0
DT0:MQWIK .BAS    15      27-FEB-74 01-AUG-73  02:22 PM 0
DT0:KEYDMP.BAS    7       27-FEB-74 01-AUG-73  02:22 PM 0
DT0:MUKRIO.BAS    3       27-FEB-74 27-FEB-74  02:22 PM 0
DT0:K     .BAS    4       27-FEB-74 26-FEB-74  02:22 PM 0
DT0:JIM   .BAS    3       27-FEB-74 26-FEB-74  02:22 PM 0
LIST COMPLETE 119 BLOCKS LISTED IN 10 FILES

#
```

BACKDK prints an information line containing the project-programmer number of the current account and the current system date and time, followed, on subsequent lines, by a list of files. For each file, BACKDK prints the full file description, number of blocks, the date of last access, the date and time of creation if applicable (if not applicable, the current system date and time), and the cluster size of the file if applicable (Ø if not applicable). For magtape and disk files, the protection code is also printed. At the end of the list of files, BACKUP prints a line containing the total number of blocks in the files listed and the number of files listed, followed by a line containing the pound sign character.

To direct the output of a directory listing to another device or to a file, the user can type the /L option, specifying the desired device or file, and the /A option. For example, the following command,

```
#/A/L:DIR.SYS
#
```

prints the directory listing of the current account on the system disk in the disk file DIR.SYS.

To verify readability of files on a device, the user can type the /V option.

```
#DT1:*.BAS/V
VERIFY COMPLETE  121 BLOCKS IN  4 FILES
#
```

BACKDK reads each block of every source file on DECtape unit 1 to ensure no errors. Upon completion, BACKDK prints an information message, followed by the pound sign character. If an error occurs, and the user requested the /L option, BACKDK records the data error, the file involved, and the block number in the file containing the error. (BACKUP treats a command without the /A option and without the left hand arrow (<) character as a verify request.

The /C:dd-mmm-yy option allows a user to transfer files based on the creation date. The option can select certain files from files of the same name on magtape based on a creation date. The /D:dd-mmm-yy option can be used to keep the system disk clear of lesser used files. The option selectively deletes files on the source device after the transfer is completed.

4.10.7 Privileged BACKUP Commands and Options

A user running BACKUP under a privileged account can specify, on the input side of a command, the project-programmer field described in Table 4-15 and can use several privileged options described in Table 4-17.

For a privileged user, BACKDK retains the date of creation and date of last access information when transferring files. The /B option causes BACKDK to reset the information to the current system date as BACKDK normally does for a non-privileged user.

The /DET option causes BACKDK to run as a detached job. The user must not specify the /I and the /L options with the /DET option since both options require printing at the terminal keyboard. However, the /L option with a filename can be specified. The /M option usually accompanies the /DET option when the output device is magtape. If /M is not specified with /DET, the job simply enters the HB state when it attempts to print a message.

The most valuable use of the privileged options is made when the user must transfer all files from the system disk to magtape, as is shown in the following command.

```
#MT1:<[*,*]/Q/M/DET/L:BACKUP.SYS/Z
```

All files in the public structure (including those in the system account Ø,1) are transferred to the magtape on unit 1. The /Z options causes BACKDK to zero the magtape before the transfer begins.

Table 4-17

BACKUP Options for Privileged Accounts

Option	Meaning
/B	Resets date of creation and date of last access information to the current system date.
/DET	Causes BACKDK to run detached from the terminal.
/M	Used with /DET option to broadcast an action message at the system console terminal (KBØ:) when a continuation reel of magtape is required or when any error message is to be printed.
/Q /Q:X	Specifies that eight 256-word blocks are to be written in (or read from) each magtape record. If an integer number between 1 and 16 inclusive is given as the argument x, BACKDK uses that number of 256-word disk blocks for each magtape record. Without /Q, each magtape record contains one 256-word block. NOTE To avoid errors, files written to magtape with the /Q option must be read from magtape with the same /Q option.
/W	Reduces file processor overhead when transferring files from a private disk pack or DECpack. The pack must not otherwise be accessed while BACKDK is running.

Since the /Q option is given, BACKDK writes eight 256-word disk blocks per record record on the magtape. Each increment of two blocks in the magtape record size causes the user job area to increase by 1K words. (To prevent the MAXIMUM CORE EXCEEDED error, ensure that a minimum of 12K words is available for the job to run.) The /Q option increases the speed of the transfer and decreases the amount of magtape needed to contain the files. Any subsequent transfer or verify operation involving magtape requires that the user specify the /Q option and the same value of x as originally specified.

When BACKDK completes the transfer, it chains back to the BACKUP program and enters the HB state. The user must attach the job to a terminal to allow BACKUP to print any messages or the pound sign character. The listing information is written to the file BACKUP.SYS.

THIS PAGE PURPOSELY LEFT BLANK.

4.11 QUE PROGRAM[1]

The QUE system program creates requests, or jobs, which are to be executed by spooling programs. QUE also executes auxiliary operations such as listing pending requests and killing pending requests. The QUE program checks the syntax and validity of each request and passes it to another program for further processing if the request is valid. If any part of a request is invalid, the program rejects the entire request, prints an appropriate error message, and allows the user to try again.

For the system to execute a request created by QUE, the system must contain two other programs: a queue manager program (QUEMAN) and a spooling program. QUEMAN processes a request created by QUE and places it in the system queue file QUEUE.SYS. The particular spooling program executes requests on its related device when the QUEMAN program passes it a pending request. Spooling programs on standard RSTS/E systems can be SPOOL, BATCH, or RJ2780. If no requests are pending in the queue file, the spooling program executes a sleep operation until activated by QUEMAN.

The file QUEUE.SYS is stored under the system library account on the system disk and occupies 262 blocks of storage. The file can accommodate a total of 250 files queued as job requests.

4.11.1 Running QUE at a Terminal

A privileged or non-privileged user runs QUE at a terminal logged into the RSTS-11 system by typing the RUN command as follows:

```
RUN $QUE
QUE  V05B-9  -  RSTS V05B-24
```

When QUE runs, it prints a header containing the program name, its version and revision numbers, and the system name and version and revision numbers. The program ensures that the system queue file exists and is accessible on the system library account. If no errors occur, QUE prints a pound sign (#) character which acts as a prompting indicator to signal the user that he can type a command. After each command is processed, QUE prints the pound sign.

[1] This feature is not available prior to Version 5B (RSTS/E) systems

If QUE runs and encounters an error when accessing the file $QUEUE.SYS, it prints the following message and terminates.

```
QUEUE NOT INITIALIZED
READY
```

The system prints READY. To run QUE without an error, the user must have the system manager or responsible system programmer run the QUEMAN program and initialize the queue file.

QUE is terminated by typing E or the CTRL/Z combination in response to the pound sign character, as shown below:

```
# E
READY
```

Control is returned to the BASIC-PLUS command level, as indicated by the READY message being printed. Commands to queue, list, and kill jobs and the related error messages are explained in the following sections. A summary of the commands appears in Table 4-18.

Table 4-18

QUE Program Commands

Command	Format	Description
Q	Q device:jobname=file spec(s)	Allows user to give a file specification(s) of a file to be processed. A comma is used to separate file specifications if more than one is given. See Table 4-17 for options which may be given with a file specification.
L	L device:=jobname(s)	Prints at the terminal a list of the currently pending requests for the spooling program. If device: is not given, LP0: is assumed. If device: is not given and jobname(s) is given, the = character is required. Valid devices are LP:, LP0:-LP7:,BA:,BA0: -BA7:, and RJ:. Specifying LP: or BA: indicates all units of that device. If no jobname appears, QUE prints all jobs.
K	K device:=jobname(s)	Removes from the queue the job(s) indicated by device specification and jobname. If device is not given, QUE assumes LP0:. If device: is not given and jobname(s) is given, the = character is required.
H	H	Causes an information message to be printed at the terminal.
E	E	Causes QUE to terminate and to return control to BASIC-PLUS command level.
CTRL/Z	Type the CTRL/Z combination	Causes QUE to terminate (same as E).

4.11.2 Using the Q Command

The Q command typed in response to the pound sign creates a request for a spooling program. The Q command has the following general format.

Q device:jobname=file spec,file spec,...,file spec

where:

device — is the device designator LP:, LP0:–LP7:, BA:, BA0:–BA7:, and RJ: and unit number of the device which the spooling program services. If a designator is not given, LP0: is assumed. If a device is specified, the = sign is required. The designations LP: and BA: cause the request to be queued for any available spooling program for that device type.

If a unit number is specified, the request is queued for that unit and is executed only if a spooling program is running on that unit. Additionally, the general batch processor BA: processes only those jobs queued for BA: and not jobs queued for BA0: through BA7:. BATCH processors BA0: through BA7: however, process requests queued for their respective units and requests queued for BA:.

jobname: — is a maximum of six alphanumeric characters to identify a user's request in the queue. If device and jobname are not specified, the = sign is optional and the filename of the first file specified in the request is the job name. Several options described in Table 4-19 may accompany the jobname.

filespec: — is the external file specification of the file to be processed from the queue and any combination of Q command options. Up to 14 files can be included in a request. File specifications are separated by a comma. (See Table 4-20 for the options.)

The jobname (or the left hand side of the = character) can have several output options which apply to the entire request. Each option must be preceded by a slash character (/). Table 4-19 lists and describes the output options.

Table 4-19

QUE Job Output Options

Option Format	Description
/MODE:n	The value n specifies the MODE which the spooling program will use in its OPEN statement for the output device. See Section 3.2 of the RSTS/E Programming Manual for information on the line printer MODE values.
/AFTER:hh:mm	The value hh:mm specifies, in hours and minutes, the 24-hour (military) time after which the queue manager will initiate the current request. For example, /AFTER:13:30 causes the job to be initiated after 1:30 p.m.
/PR:n	QUE sets the priority to the value n which can be between 0 and 128. If /PR:n is not specified, QUE assigns the value 128. A privileged user can specify a priority between 0 and 255.
/FORMAT:nn	Use the format specified by nn for printing the file. Value may be: FO FORTRAN forms control. NO Embedded forms control (equivalent to papertape) LF Implied forms control (LF and CR printed before each record) If the input device is not magtape or is DOS/BATCH format magtape, the default assumes embedded forms control. If the input device is ANSI format magtape, the default is the forms control denoted in the magtape label.

The external file specification of a file to be included in the job can have the following form:

device:filename.extension[proj,prog]/option/option...

A valid device designator is either DKn:, DPn:, MTn:, DTn:, or null (which indicates the system device in the public structure). QUE interprets logical names if the user makes the assignment before QUE runs. Otherwise, an unassigned logical device name generates the ILLEGAL INPUT FILE error. See Section 2.7 for a description of logical names.

The filename and extension may consist of ? and * characters described in Table 4-15 in Section 4.10 of this guide. The [proj,prog] designation is the project-programmer number (account) under which the file to be printed is stored. The designation can be omitted if the file is stored under the current user's account. The /option designation is a slash character (/) followed by one of the Q command options. The options are described in Table 4-20.

Table 4-20

Q Command Options

Option	Format	Description
Copies	/C:n	Tells SPOOL to print the number of copies indicated by the decimal integer n. If /C:n is not given, one copy is printed.
Noheader	/N	Tells SPOOL to suppress printing of the file header.
Delete	/D	Tells SPOOL to delete the file after it is printed if the protection code of the file permits. If /D is not specified, the file is left intact. The QUEMAN program checks the protection code of the file.
Restart	/R:n	If a system or device malfunction occurs while SPOOL is printing the file, backup n physical disk blocks from the point at which printing stopped and continue printing. The value n can be a decimal integer between 0 and 255. A value of 255 restarts the job from the job header page onward.
	/R	Equivalent to /R:255. SPOOL resumes printing from the job header page onward. If /R:n is not given, SPOOL uses /R:0.
More	/MORE	Used at the end of a command line only to indicate to QUE that the user has more text to type on the next physical line. If used when chaining to QUE the program generates error 12.
Binary	/B	Used to indicate a binary file to the RJ2780 program, which must be in TRANSPARENT mode.

The following command typed to QUE,

```
#Q ABC.DAT
#
-
```

causes the file ABC.DAT stored under the current user's account on the public structure to be sent to the queue manager for printing on the spooled line printer, unit 0. SPOOL generates one copy of the file under the user job header ABC. If printing of the file is interrupted for any reason, the program continues printing after the last character printed before the interruption.

To queue a file from a device other than the public disk, specify the device designator in the command. For example,

```
#Q DT0:A.BAS/C:2
#
```

The indicated file on DECtape unit 0 is queued for printing on line printer unit 0. The device must not be assigned or otherwise in use since the system generates the DEVICE NOT AVAILABLE error when SPOOL attempts to access the file. SPOOL generates two copies of the file under the job header A.

To specify a job name for the request, type the name and the = character in the command. For example,

```
# Q SORT=DK1:FILEA,DK1:FILEB/C:2
```

SPOOL prints SORT in the job header and generates one copy of FILEA and two copies of FILEB.

To specify a spooling program other than the one for line printer unit 0, type the appropriate designator and the = character in the command. For example:

```
# Q LP1:=DT0:FILOUT.001/R:2
#
```

QUE creates a request for FILOUT.001 on line printer unit 1, with job name FILOUT. If printing is interrupted, SPOOL backs up two blocks when it resumes printing.

To specify a request longer than one physical line, type the /MORE option as the last item on the line. For example,

```
#Q LP1:PRINT1=FILE1.001,ABCDEF.001,/MORE
MORE > HELP.TXT,ACCT.SYS
#
```

As a result, QUE prints, as a prompting indicator, the text MORE>, after which the user can continue typing the request. QUE allows up to fourteen files in one request. The user can type /MORE as many times as necessary.

When a device designator is specified in the command, QUE uses that device (and not the system device) as a default for subsequent input file names on the physical line. For example, when this command is executed:

```
#Q LP1:SORT=DT2:FILEA,FILEB
```

QUE sets the device to DT2: for both FILEA and FILEB. Because no device designator is specified for FILEB, DT2: becomes the default device.

To set a priority to a job when queuing a file, the user specifies the /PR:n option with a value between 0 and 128. Without the /PR option, QUE assigns a priority of 128 to the job. The queue management program processes jobs with a priority of 128 or greater before processing jobs less than 128. Only privileged users can specify a priority greater than 128. With such a priority scheme, the queue manager can process important jobs before routine jobs and can delay processing less important jobs until routine jobs are completed.

4.11.3 Using the L Command

The L command lists, at the user's terminal, information concerning pending requests for a specified device. For example:

```
#L LP1
LP1 QUEUE LISTING        Ø3-JAN-75        Ø9:58 AM

UNIT   JOB              S / P    FILES
 1     ADDRES[12Ø,71]   1 / 128   SY   :ADDRES.      [ 120 , 71 ]
 1     MASTER[120,71]   Ø / 128   SY   :MASTER.BAS   [ 120 , 71 ]
 1     ADRES [12Ø,71]   Ø / 128   SY   :ADRES .      [ 12Ø , 71 ]
#
```

The L command causes QUE to print two header lines for the specified spooled device. The first header line contains the device and unit designator of the spooled device, the current system date, and the time of day. The second header line contains text to denote the unit queued to, the job name and account number of the requester, the status, priority, and full file specifications associated with each job. If no device and unit designator appear in the L command, QUE prints a listing of line printer unit 0 jobs. If no jobs are queued, QUE prints the header lines followed by the # character. Otherwise, jobs are listed in the order which QUEMAN accesses them. The status indicates whether the job is waiting to be processed (value Ø) or is being processed by the spooling program (value of non-zero). To list jobs on all line printers, type the L LP: command. QUE prints the related unit number in the UNIT column for each queued job. Jobs queued for the BATCH processor are listed by typing the L BA: command; jobs queued for the RJ2780 program are listed by typing the L RJ: command.

To list only information concerned with one request or requests, include the = character followed by the jobname(s) in the L command. For example,

```
#L LP1:=ADRES
LP1  QUEUE LISTING          Ø3-JAN-75        Ø9:59  AM
UNIT     JOB                S / P      FILES
 1       ADRES [12Ø,71]     Ø / 128    SY    :ADRES .    [ 12Ø , 71 ]
#
```

QUE prints the heading lines and the data for the job name specified. If the user specifies more than one jobname, each must be separated by commas. For example,

```
#L LP1:=ADRES,MASTER
LP1  QUEUE LISTING          Ø3-JAN-75        1Ø:Ø1  AM
UNIT     JOB                S / P      FILES
 1       MASTER[12Ø,71]     1 / 128    SY    :MASTER.BAS   [ 12Ø , 71 ]
 1       ADRES [12Ø,71]     Ø / 128    SY    :ADRES .      [ 12Ø , 71 ]
#
```

4.11.4 Using the K Command

The K command removes a job or jobs from the queue file QUEUE.SYS for a specified device. To remove a job, specify the device designator and the = character followed by the job name or job names separated by commas. For example,

```
#K LP1:=PRINT1,BATCH1
#
```

As a result, each job for the user's account in the LP1: part of the queue with the names PRINT1 or BATCH1 is deleted if it is not currently being processed by the LP1: spooling program. If the spooling program is currently processing a file for a job killed by QUE, the queue manager program QUEMAN completes that file and subsequently terminates processing on the remainder of the job. To immediately kill a job which is currently processing, the user must generate an error to interrupt the spooling program (for example, by putting the line printer unit off line) and must type an appropriate command to terminate processing by the spooling program on that job. Section 5.2 of the RSTS/E System Manager's Guide describes the SPOOL commands.

4.11.5 Chaining to QUE from a User Program

To run the QUE program by a CHAIN operation, the user program must first put in core common a string in the following format.

Program name	The filename specification of the program to which QUE passes control. The specification can include a project-programmer field but not an extension. This program need not be the calling program.
Delimiter	The delimiter must be CHR$(13), the CR character.
Line number	An integer number in CVT%$ format indicating the line number of the program to which QUE passes control. For example, CVT%$ (28000%).
Command	The string QUE executes. It must have the same syntax as if it were typed at the terminal in response to the # character but must not include CR and LF characters. The option /MORE must not be used.
Delimiter	The delimiter must be CHR$(13).
Text	Optional string which QUE places in core common when it completes processing and passes control. Its length is restricted by the number of bytes remaining in core common.

Secondly the user program must execute a CHAIN "$QUE" 31000 statement, to actually pass control to QUE stored in the system library account.

Upon completing the request passed to it through core common, QUE places a string in core common and executes a CHAIN "program name" line number statement. The program name and line number are taken from the values passed to QUE in core common. The core common string QUE passes has the following format.

Error number	The QUE error code in CHR$ format. For example, CHR$ (E%). See Section 4.11.6 for a description of the error codes generated by QUE. CHR$ (0%) indicates no error code was generated.
Text	The optional string which the user program passes to QUE.

4.11.6 Error Messages and Codes

QUE reports an error condition by printing a message or by returning an error code. When running at a terminal, QUE prints a unique message which describes the error condition. No part of the command typed is executed. The user must retype the entire command in the correct format. Any RSTS-11 system errors encountered are reported in the ILLEGAL INPUT FILE message. When running by means of a CHAIN operation from a user program, QUE reports an error by an error code when it passes control. Table 4-21 gives both the messages and codes and describes the related error conditions.

Table 4-21
QUE Error Messages and Codes

Error Code	Message	Meaning
0	None	QUE processed command without error.
1	INVALID COMMAND-<text>	The<text>indicated is an undefined command.
2	SW ON WRONG SIDE OF COMMAND	The option switch specified in the command line to QUE is reserved for the input side and was found on the output side, or vice-versa.
3	INVALID SWITCH FORMAT	The option contains an invalid request.
4	UNDEFINED SWITCH	The command line contains an undefined option switch.
5	TOO MANY FILES	No more than fourteen files can be given in the Q or K command.
6	NULL FILE SPEC	At least one file specification must be given in a Q or K command and none was found; or two commas occurred with no file specification between them.
7		Not used.
8	ILLEGAL INPUT FILE - <text--file spec>	The request is invalid because the file indicated by the file specification caused the RSTS error described in the text. The user must retype request.

Table 4-21 (Cont.)
QUE Error Messages and Codes

Error Code	Message	Meaning
9	WILDCARDS NOT ALLOWED IN PPN	An asterisk is not allowed in the project-programmer field.
10	QUEUE FULL	The queue file is temporarily full. Try again when the spooling program has processed some pending requests.
11	BAD SWITCH VALUE - text	QUE encounters an invalid or improper number in the option indicated by the text. For example, user specifies /MODE:A or a non-provileged user specifies /PR:129 or greater.
12	'MORE' REQUESTED ON A CHAIN	The /MORE option is not allowed when a program runs QUE by a CHAIN operation.
13	NOT A QUEUABLE DEVICE	The user attempted to queue a request to a device which cannot handle queued requests. For example, Q LG:=JOB.
14		Not used.
15	QUEUE NOT INITIALIZED	The queue file QUEUE.SYS does not exist in the system library account. A privileged user must run QUEMAN to initialize the queue.
16	QUEMAN NOT RUNNING - CAN'T QUE OR KILL	User attempts to queue or kill a job and, because queue manager is not running on the system, QUE generates an error. However, user can list jobs without QUEMAN running.

4.11.7 Running QUE by CCL Commands

On RSTS/E systems not modified otherwise, a user can run QUE by means of the concise command language (CCL) command QUE. The correct format is QUE/ command line where command line is any valid QUE program command. If the QUE program does not find a / character immediately following the characters QUE, it defaults the command to a Q command. For example,

```
QUE/Q JOB1 = ABC.DAT
```

and

```
QUE JOB1 = ABC.DAT
```

The commands are equivalent since both enter JOB1 in the queue for line printer Ø. In either case, the QUE program runs and executes the command. If no error is detected, the system prints the READY message. An error in the command causes QUE to print the related error message on a subsequent line, after which the system prints READY.

To include an output option when queuing a job, the user must begin the QUE command with /Q or else must give an explicit job name in the QUE command. For example,

```
QUE/Q/MODE:2Ø48=DTØ:DISK.BAS
READY
```

The / character preceding the Q character differentiates the Q command from the job name Q.

To list or kill jobs, the user must type QUE, followed by a slash (/) character and the appropriate command. For example,

```
QUE/L LP2:=PRINT1
LP2 QUEUE LISTING    19-JUN-74    Ø4:33 PM
UNIT JOB             S/P     FILES
 2   PRINT1[12Ø,8Ø]  1/Ø     DTØ:DISK.BAS[12Ø,8Ø]
READY
```

QUE runs and executes the command indicated following the / character, after which control is returned to BASIC-PLUS command level.

4.11.8 Running QUE at a Logged Out Terminal

The currently pending requests for a spooling program can be printed by typing the QUE/L dev: command. For example, to print the line printer unit 1 queue, type the following command.

```
QUE /L LP1:
```

QUE executes the L (LIST) command for the LP1: queue. After printing the pending requests, QUE exits to RSTS/E and prints BYE.

4.12 BATCH PROCESSING[1]

The capability to execute a batch of commands allows the user to submit jobs to be run without terminal dialogue. Batch processing is particularly useful in executing large data processing operations for which interactive requirements are not a factor.

Batch input can be submitted from standard files on a random access device. For purposes of this description, input is dealt with as though it were on cards, where each card represents one record. Such input consists of elements of the batch control language and is collectively referred to as a batch stream. It is possible to execute multiple streams simultaneously by running multiple copies of the BATCH program. The capability to run more than a single batch stream is controlled by the system manager.

Sections 4.12.1 through 4.12.3 discuss the elements of the batch control language. Operating procedures are described in Section 4.12.4.

4.12.1 Control Statements

Batch control language statements consist of a command field, specification field(s) and a comment field, in the following format:

$<command-field> [specification-field(s)] [!comment]

Fields must be separated by one or more spaces and/or tabs.

A command field must always be present and may contain switch modifiers to control or limit the command. When appropriate, the command field is followed by one or more specification fields. The ! character is a comment prefix signifying that the information between the ! character and the line terminating character consists of comment information. The system takes no action on comment information.

The dash (-) character can indicate a continuation of a command. If a dash is the last character in a command, the next line is treated as a continuation of the previous line and must begin with a $ followed by a blank.

[1] This feature is not available prior to Version 5B (RSTS/E) systems.

Double quote characters may be used in control statements to reproduce some text identically and override any special interpretation of characters by BATCH. For example, the exclamation point (!) in RSTS/E is the designator for the auxiliary library account [1,3] or for an installation defined account. In BATCH, the exclamation point signifies a comment. To prevent BATCH from misinterpreting the ! character given as an account designator, use quote characters as shown in the following sample control statement.

```
$RUN "!UPDAT"
```

As a result, BATCH executes the program UPDAT from the auxiliary library account. Without the quotes in the above example, the current program is executed and the characters following the ! character are treated as comment characters.

4.12.1.1 Command Field - The command field comprises the following elements:

1. A $ (dollar sign) character is the first character position. The $ character is the control statement recognition character.
2. The command name begins in the second character position and immediately follows the $ character. For example, $JOB. No blanks are allowed in the command field because a blank (or horizontal tab) delimits the command field.
3. Valid switches. No blank can appear between the command name and a switch. Switches are denoted by a / character. For example, /NAME=COMPL.

NOTE

Command names and switch names can be shortened to their first three characters. For example, the system interprets the command BAS as well as BASIC. Any other form, such as BASI, is invalid.

Multiple, adjacent blank characters are equivalent to one blank character. A horizontal tab is equivalent to one blank character. A blank character delimits the command field; otherwise the blank character is ignored.

4.12.1.2 Specification Fields - A list of specification fields immediately follows the command field delimiter. The following rules apply:

1. Specification fields are separated by blanks.
2. Specification fields are terminated by a ! character if followed by a comment, or are otherwise terminated by a line terminating character.

3. Depending on the command, a specification field consists of a device specification, a file specification, or an arbitrary ASCII string, any of which can be followed by appropriate switches. The / character signals the start of a switch. For example, XYZ.BAS/SOURCE. The switch indicates that the file is a source file.

4.12.1.3 Comments - The following rules govern comment fields.

1. The start of a comment is defined by an ! character in the control statement.
2 Any character following an ! character and preceding the end-of-line terminator is treated as a comment and is otherwise ignored by the batch processor. Comment lines with no text may force line spacing on the job log and thereby make the log more readable. To force line spacing, simply include lines consisting solely of $! followed immediately by a carriage return/line feed.

4.12.1.4 Syntactical Rules - The following are syntax rules for control language statements.

1. A control statement must have a command name (except in the case of the comment line "$!"). If the command name is omitted, the command is ignored. An unrecognizable command name is illegal. An appropriate error message is printed by the batch processor. Only two forms of the command are recognized: the full name and the 3-character abbreviation of the name. For example, the following are the legal BASIC-PLUS commands:

   $BASIC $BAS

   Switches in the command field apply to the entire command. If a switch appears in the specification field which contradicts a command field switch, an error results.

2. An asterisk is allowed in the filename or the file type field of a file specification. See Section 4.12.2 for the description of file specifications. An asterisk can refer only to files already created. An asterisk appearing in a file specification used in file creation constitutes an invalid file specification.

   The batch processor uses the leftmost 6 characters from file name fields longer than 6 characters and uses the leftmost 3 characters from the file type fields longer than 3 characters.

   An invalid filename field, file type field, device specification, or project-programmer field in a file specification causes the control statement in which it appears to be invalid. The batch processor does not execute an invalid control statement. An error message is printed.

3. Switches can be used in the command field and specification fields of a control statement. Switches appearing in the command field are command qualifiers, and their function applies to the entire command. Switches appearing in specification fields apply only to the field in which they appear.

   Unrecognizable switches invalidate the control statements in which they appear.

4.12.1.5 Syntax Example - The following are sample control statements which illustrate the syntax of Batch statements.

```
$JOB/NAME=SMYTHE !FIRST JOB
$!
$!COMPILATION OF NEW SOURCE FILES
$!
$MESSAGE STARTING COMPILATIONS
$BASIC XYZ/SOURCE XYZ.LIS/LIST XYZ/OBJECT
$BASIC ABC/SOURCE ABC.LIS/LIST ABC/OBJECT
$!
$MESSAGE STARTING LISTING OUTPUT
$!
$PRINT *.LIS! ALL LIST FILES
$!
$EOJ
```

4.12.2 File Specifications

A file specification appears in the specification field and is an alphanumeric string containing the following elements:

filename.type

The batch processor assigns default values if part of the file specification or all of it is optionally omitted.

4.12.2.1 Filename Specification - A filename specification is a string of alphanumeric characters of which the first six characters must be unique. An asterisk in the filename field denotes all files of the specified type in the account designated. If necessary, the batch processor generates a default filename related to the time of day as described in Section 4.12.2.3.

4.12.2.2 File Type Specification - A file type specification consists of a period, immediately followed by a string containing three or more alphanumeric characters, the first three of which must be unique.

The file type reflects the type of the file. For example, a BASIC source file has .BAS as its file type. An asterisk in the file type field denotes all file types including files with no type specified.

Standard file types are listed below.

.CTL	Batch control file
.DAT	Data file
.DIR	Directory file
.BAS	BASIC source file
.LIS	List file
.BAC	BASIC compiled output file
.CBL	COBOL source file
.OBJ	COBOL compiled output file
.SRT	PDP-11 SORT11 input, output or listing file

These file types are the defaults when no file type is specified. The default chosen is determined by the current operation and by the type of file expected. Table 4-22 summarizes the default file types that apply to particular batch commands.

Table 4-22

Batch Commands - Related Default File Types

Command/ Section	Default Type	Meaning
$BASIC 4.12.3.3	.BAS	Input source file default type.
	.BAC	Output object file default type.
	.LIS	Listing file default type.
$CREATE 4.12.3.4	.DAT	The file generated as output by CREATE has a file type of .DAT.
$DIRECTORY 4.12.3.4	.DIR	The file in which the directory is to be recorded has a file type of .DIR.
$JOB 4.12.3.1	.CTL	Batch control file default type; assumed when the Batch job is on a file-structured device.
	.LOG	Batch output log file default type.
$PRINT 4.12.3.4	.LIS	Default type of file to be printed.
$RUN 4.12.3.5	.BAC	Default type of file to be run.
$COBOL[1] 4.12.3.11	.CBL	Input source file default type.
	.OBJ	Output object file default type.
	.LIS	Listing file default type.
$SORT[1] 4.12.3.12	.SRT	Input, output or listing files default type.

[1] This feature is not available before Version 6A systems.

4.12.2.3 File Specification Defaults - Defaults are assigned to omitted filename and file type elements as shown in Table 4-23.

Table 4-23

File Specification Defaults

Condition	Default	Example
Name specified but no file type	Default assigned as appropriate to the current operation. For example, with the BATCH command BASIC, the default is .BAS.	ABC=ABC.type
Name,followed by dot, but no file type	Default file type is null. No file type is assigned.	ABC.=ABC
File type, but no name	Default filename is related to time of day.	.LIS=B2347P.LIS (created at Ø1:23:47 PM)
No file specification	Default filename (related to time of day) with default type as appropriate to current operation.	Null=B2347P.type

4.12.2.4 Switch Specification - Switches consist of a / character followed immediately by a name. If the switch takes an argument, the argument is separated from the switch name by a colon (:) or equal sign (=). If the switch takes an argument and subarguments, each sub-argument is separated from the argument and themselves by a colon. For example,

```
/NAME=JOB3
/VID="MY TAPE"
```

Switches accept arguments of standard types, such as decimal constant, alphanumeric string, and date-time.

Switch values can be negated by putting the characters NO between the / character and the switch name. For example,

```
/NOOBJ
```

This switch indicates that no object file is to be produced. It would be used in conjunction with the $BASIC and $COBOL commands, usually.

NOTE

The negation characters NO are not considered part of the switch name. Thus, a negated switch must contain at least five characters. For example:

/NOOBJ or /NOOBJECT

is valid, but

/NOO

is invalid.

4.12.3 BATCH Commands

The BATCH command set consists of

$JOB	- begins a job
$EOJ	- ends a job
$BASIC	- executes BASIC-PLUS compiler
$utility	- executes system utility function
$RUN	- executes program
$DATA	- begins data images
$EOD	- ends data images
$MESSAGE	- logs message on control terminal
$MOUNT	- assigns device
$DISMOUNT	- deassigns device
$COBOL[1]	- executes the COBOL compiler
$SORT[1]	- executes the PDP-11 Sort program SORT11

4.12.3.1 $JOB - This command marks the beginning of a job. The following command switches are allowed.

/NAME=jobname	This switch assigns a name to the job. Job names can be up to 6 characters long. This name overrides the control file name as the identifier of the job.
/NONAME	This switch indicates that no job name is defined. A default job name is assigned. The default job name is the name of the control file. The name appears in all messages to the system operator.
/LIMIT=nnn	This switch is used to assign an elapsed time limit to the job. The value of nnn, a decimal number, is interpreted as minutes.

[1] This feature is not available before Version 6A systems.

/NOLIMIT[1] — Gives the job an unlimited amount of elapsed time to complete. If neither /LIMIT:nn nor /NOLIMIT appear, the job is given 10 minutes elapsed time to complete execution before BATCH terminates it.

/PRIORITY:n[2] — Sets the RSTS/E job priority to n (or the nearest multiple of 8) for the BATCH stream. For privileged users, n can be between -127 and +127; for non-privileged users, n is limited to a value between -127 and -8. Unless otherwise altered by the /PRIORITY:n switch, all jobs run at -8 priority.

/CCL — This switch allows the use of the system's interactive Concise Command Language. When this switch is specified, any of the system commands which do not conflict with existing Batch commands may follow the $ character. The batch processor insures that the job is in the READY state before executing the command.

The following specification field may be included:

[n,m] — To have the job executed on an account other than that under which it was queued, a specification field may indicate the account number desired. This can only be used by a privileged user.

The following are the error conditions possible.

Unrecognized switch
Illegal switch value
Multiple conflicting specifications (switches)
Different account specified by non-privileged user.

4.12.3.2 $EOJ - This command marks the end of a job. The $EOJ command automatically dismounts all devices mounted by the job. $EOJ prints an appropriate message to the operator that the logical device should be dismounted. A logical deassignment is performed.

NOTE

a. A $EOJ is implied when a physical end of file condition or a $JOB control statement is encountered while processing a job file.

b. No switches are legal on the $EOJ command.

The following are the error conditions possible.

Non-comment characters following $EOJ.

[1] On systems before Version 6A, the /NOLIMIT switch set the time limit to 10 minutes.

[2] This feature is not available before Version 6A systems.

4.12.3.3 $BASIC - The $BASIC command calls the BASIC-PLUS compiler, which compiles the source program into an object program. The format of the $BASIC command is:

$BASIC[switches] [specification fields [switches]]

The following command field switches are valid.

/RUN	Compile and execute the program.
/NORUN	Compile only. If no switch is specified, only compilation occurs.
/OBJECT	Produce a compiled file.
/NOOBJECT	Do not produce compiled file. If no switch is specified, a compiled file is not produced.
/LIST	Produce a listing file.
/NOLIST	Do not produce a listing file. If no switch is specified, a listing file is not produced.

As indicated, the specification field is optional. Up to three file specifications can appear: source, compiled, and listing. These are indicated by specifying the appropriate switch:

/BASIC	
or	indicates input source file
/SOURCE	
/OBJECT	indicates output compiled file
/LIST	indicates output list file

If a source file is not specified, the $BASIC command must be followed by a set of BASIC-PLUS source statements, terminated by either $EOD (see Section 4.12.3.7) or some other recognized batch control statement. For example,

```
$BAS            LISTING/LIS
    BASIC-PLUS
    Source
    Deck
$EOD
```

If a source file is explicitly specified, any source statements following this command are appended to the source program. Source statements following this command and having line numbers equal to those in the source program replace those in the source program. Source input must be provided, either through a file specification, or through source statements, or both.

If no listing file is specified, but the /LIST switch is present, the Batch processor creates, prints, and subsequently deletes the default listing file. If a file specification appears with the /LIST switch, the batch processor does not automatically queue the file for printing. To print the file specified as part of the batch job, the user must supply a $PRINT control statement described in Section 4.12.3.4.

If no object file is specified with the /OBJECT switch, a default object file is created and is deleted after job completion. If an object file is explicitly specified, it is preserved after job execution. Errors result from conflicting switch specifications such as both /BASIC and /SOURCE on different specification fields.

The default applied when a file is specified without a switch is /SOURCE.

The following are the error conditions possible.

Unrecognized switch
Multiple conflicting specifications (switches)
File specification syntax error

4.12.3.4 Utility BATCH Commands

The following utility functions are provided by the RSTS/E batch processor.

$DELETE	-	deletes files
$COPY	-	copies files
$PRINT	-	prints a file on the system line printer by means of the line printer spooling program SPOOL.
$DIRECTORY	-	lists a file directory
$CREATE	-	creates a file from data in the input stream

$DELETE

The $DELETE command is used to delete specified files. It is specified in the following format.

$DELETE file1 [file2 ... filen]

The filename and file type must be included. An asterisk is not valid in either the filename or the file type field.

No switches are used with $DELETE.

The following are the error conditions possible.

No file specification
Syntax error in file specification

$COPY

$COPY is used to copy files. Use of the asterisk character in the filename and extension specification is invalid. The following are the valid switches.

/OUTPUT	for new files to be created
/INPUT	for files to be copied

The following is an example of the $COPY command.

```
$COPY TER.LIS/OUTPUT TERRY.LIS/INPUT
```

The $COPY command supports the use of + (plus sign) to indicate file concatenation. When used with $COPY, file concatenation results in the creation of a single file, consisting of files connected together. The + character appears in the file specification field, between the specifications of files to be concatenated. If no switch is specified, /INPUT is assumed.

The following are the error conditions possible.

No output specification
No input specification
Multiple conflicting specifications
Syntax error in file description

$PRINT

The $PRINT command prints the contents of files on the system line printer by means of the spooling program SPOOL. No switches are valid. Asterisks in file specifications can be used. The $PRINT command is formatted as follows.

```
$PRINT file1  [ file2 ... filen]
```

where the specification field contains the file or files to be printed.

The following are the error conditions possible.

No file specification
Syntax error in file specification

$DIRECTORY

$DIRECTORY produces a directory listing of the file(s) in the specified account and has the following format:

```
$DIRECTORY [specification field]
```

The specification field can contain file specifications. If no file specification appears, the $DIR command lists the contents of the current account on the batch log device. A file specification indicates the directory of a file or set of files and can contain an asterisk in either the filename field or file type field. For example,

```
$DIR *.BAS
```

The statement creates a directory listing of all files in the current account with the .BAS extension.

To create a directory in a disk file rather than on the batch log device, the user can specify a file and the /DIRECTORY switch. For example,

```
$DIR BAJOB.DIR/DIR
```

creates the directory listing in a file BAJOB.DIR on the system disk under the current account.

To create a directory in a disk file and to designate which files are to be listed, the user must specify both the /DIRECTORY and /INPUT switches with the related file specification. For example,

```
$DIR BA.DIR/DIR *.BAC/INPUT
```

The $DIR command shown subsequently creates a directory listing of all compiled BASIC-PLUS files and stores the listing in the file BA.DIR on the system disk under the current account.

The possible error conditions are:

Syntax error in file specification
Multiple conflicting specifications.

$CREATE

The $CREATE command creates a file as indicated in the specification field. The file consists of the data images following the $CREATE command in the input stream. Data images must follow $CREATE and must be terminated by $EOD, or an error occurs. The data images must not be preceded by any other command because the $CREATE function terminates on encountering a $ in the first column of a data image.

Any previously existing file of the name specified is deleted at batch execution time, and replaced by the file created by the $CREATE command.

The $CREATE command has the following format.

```
$CREATE file
```

The following are possible error conditions.

Syntax error in file specification
No file name specified
Non-comment characters following file specification

4.12.3.5 $RUN - The $RUN command causes execution of system programs. For example, to run PIP, specify

```
$RUN $PIP
```

followed by appropriate PIP commands. The PIP program reads the commands as data images in the input stream. Execution of PIP is terminated when the next batch control statement is read.

No switches can be specified. The general format of $RUN is:

```
$RUN [file]
```

where file specifies the executable program. If file is omitted, the default current program is used.

The following are the possible error conditions.

Syntax error in file specification
Non-comment characters following file specification

4.12.3.6 $DATA - The $DATA command permits the user to include source data in the input stream following a $BASIC command (with the/RUN switch) without specifying a filename for the source data. The $DATA command is specified in the following manner.

```
$DATA
```

No specification field or switches are recognized. If no data records follow $DATA, a zero-length file is implied .

Possible error conditions are the following:

Non-comment characters between $DATA and the line terminator.

4.12.3.7 $EOD - $EOD marks the end of data records included in the input stream following commands such as $BASIC, $CREATE, $DATA, and $RUN. For example,

```
$DATA
  .
  .
  .
data
  .
  .
  .
$EOD
```

4.12.3.8 $MESSAGE - This command logs a message on the system control terminal. It provides a way for the job to communicate with the operator. The command has the following format.

```
$MESSAGE[/WAIT] message-string
```

The /WAIT switch can be used in the command field to indicate a pause to wait for operator action. The system pauses until the operator gives the appropriate command. For example,

```
$MESSAGE/WAIT MOUNT SCRATCH TAPE ON DTØ
```

The operator then mounts the tape and responds with CO to continue batch processing. If the operator finds it necessary to kill the job or to defer it for later execution, he does so by typing one of the following responses:

RE - restart
KI - kill
DE - defer

RE results in the log message JOB RESTARTED BY OPERATOR and the job starts over again from the beginning of the Batch control file(s).

KI prints the message JOB KILLED BY OPERATOR on the log and terminates the job.

DE causes the message JOB DEFERRED BY OPERATOR and places the job at the end of the Batch queue. An invalid response results in a second request for a valid response from the operator.

4.12.3.9 $MOUNT - The $MOUNT command causes a mount message to be printed on the control console, and effects a logical to physical device assignment. The physical device refers to a physical device type. The operator responds with the drive number. An automatic /WAIT occurs. Logical device names of up to six characters are used to specify logical devices.

The $MOUNT command is specified in the following format.

$MOUNT devn:[/switch] devm:[/switch]

Both the logical device and the physical device must be specified. The colon is required as the terminator for each device specification. The following switches can be used for the physical device.

/PHYSICAL	identifies the device specification to be the physical device (default)
/WRITE	tells the operator to write-enable the device (or volume)
/NOWRITE	tells the operator to write-protect the volume (default) (not permissable for RSTS/E disks.)
/VID	identifies the volume for the operator

The following switch is used with the logical device:

/LOGICAL	identifies the device specification to be the logical device name.

The /VID switch on the physical device field is used to specify the volume identification. The value associated with /VID is the name physically attached to the volume. It is included to help the operator locate the volume.

The string specified for /VID can be delimited by quotes, if the name must contain blanks. For example,

/VID="FJM JT"

No blanks are allowed in a string not delimited by quotes. For example,

$MOU M7:/PHY/VID="MY TAPE" TAPE:/LOG

In this example, logical device name TAPE is assigned to a 7-track magnetic tape unit. The operator is told that the reel of tape to be physically mounted is labeled MY TAPE. He then responds with the physical unit number on which the tape is mounted. Thereafter, in the batch command file, reference to the device TAPE: accesses the physical device on which the operator mounted the reel MY TAPE. If the physical device is a removable disk pack or cartridge, the logical device name must be the pack identification. The batch processor logically mounts and unlocks private disks which the operator mounts as a result of $MOUNT.

The valid physical devices that can be requested for mounting are:

CR:	Card Reader
DK:	Disk Cartridge
DP:	RPII-C/RP03 or RP02 disk pack
DB:	RHII/RP04 disk pack
DX:	RX01 floppy disk
DT:	DECtape
LL:	Line printer with lower case
LP:	Line printer
LU:	Line printer with Upper case only
M7:	7-track Magtape
M9:	9-track Magtape
MT:	Magtape
PP:	Paper tape punch
PR:	Paper tape reader
PS:	Public Storage (equivalent to SY:)
SY:	System device
TT:	Teletype (or terminal)

The following are possible error conditions.

Syntax error in device specification fields
Invalid device name/unit
Invalid logical device name specifications.
Unit number already assigned

4.12.3.10 $DISMOUNT

The $DISMOUNT command causes the logical to physical device assignment effected by the $MOUNT command to be nullified. It also prints an operator message, requesting that the volume be dismounted. If a /WAIT switch is included in the command field, the job will not resume until a response, as with the MESSAGE command, is received from the operator. For example,

$DIS/WAI TAPE:

The following are possible error conditions.

Syntax error in specification field

Illegal switches

Logical device not assigned

4.12.3.11 $COBOL[1] - The $COBOL command calls the COBOL compiler which compiles the source program and generates an object program. The format of the command is as follows.

$COBOL[switches] [specification fields [switches]]

The following command field switches are valid.

/RUN	Execute the previously compiled object file. Only an object file can be specified.
/NORUN	Compile the source program but do not execute the object. If /NORUN is omitted, the source program is compiled and executed.
/OBJECT	Produce a compiled file. If neither /OBJECT nor /NOOBJECT appears, /OBJECT is used.
/NOOBJECT	Do not produce an object file. If neither /OBJECT nor /NOOBJECT appears, /OBJECT is used.
/LIST	Produce a listing file. If neither /LIST nor /NOLIST appears, /LIST is used.
/NOLIST	Do not produce a listing file. If neither /LIST nor /NOLIST appears, /LIST is used.
/MAP	Include the DATA division map in the listing file.
/LOD	Create the file nnnnnn.MAP (where nnnnnn is the source file name) to contain the program load map.
/CVF	Source code is in conventional format.
/ACC:n	Accept errors in the source code of severity n or less.
/ERR:n	Suppress the printing of diagnostic messages if error severity is less than n.
/USW:n:n ...	Set run time user switches for the compiler. Switch values must be separated by colons. The range of values is 1 through 16.
/HELP	Print a help message in the log. No other switches and no file specifications are permitted with the /HELP switch.

[1] This feature is not available prior to Version 6A (RSTS/E) systems.

If neither /RUN nor /NORUN appears in the command field, COBOL automatically compiles the source program and executes the object program. If either a source file or a listing file or both are specified with /RUN, a conflict occurs. The /RUN switch indicates that an object file is to be executed. Source and listing files are specified only when a compilation is performed. BATCH does not report the file specification(s) as errors but does not pass them to the COBOL compiler for processing.

A maximum of 3 file specifications can appear in the specification field. Each specification can be differentiated by one of the following switches.

/COBOL or /SOURCE	Indicates input source file. If specification has no switch, /COBOL is used for that file.
/OBJECT	Indicates output (compiled) file.
/LIST	Indicates output listing file.

If /OBJECT does not appear in the specification field, BATCH creates a default object file and deletes it after the program run is completed. If a listing file does not appear in the specification field, but the /LIST switch is included, a default listing file is created, printed, and deleted. Explicitly specified output files (both object and listing) survive the execution of the batch job that created them. If a listing file appears in the specification field, BATCH creates the listing file but does not automatically queue it for printing. To print the listing file as part of the BATCH run, a $PRINT control statement must be supplied. (Section 4.12.3.4 describes $PRINT.)

If a file specification appears without a switch, BATCH uses the /COBOL switch for that file.

If a source input file specification does not appear in the $COBOL control statement, source statements must immediately follow and must be terminated by either an $EOD statement or some other BATCH control statement. For example,

```
$COBOL/MAP/LOD  COB.LST/LIS

    COBOL
    Source
    Deck

$EOD
```

Because the command field contains neither /RUN nor /NORUN, BATCH assumes a compilation and execution is to be done. A source file is not specified and the data following the $COBOL command is compiled.

The following command and text describe the defaults used in the $COBOL command.

```
$COBOL FILE
```

A compilation and execution is done. Because the specification FILE contains no switch, /COBOL is used. The default type CBL is used. The source program FILE.CBL is compiled and a default object file is created and executed. The object file is automatically deleted before the next command is executed with one exception. If the next command is $DATA, the object file is deleted after the end of data and before the command following the data. A default listing file is created and automatically queued for printing. The listing file is deleted after it is printed.

The following are the error conditions possible.

Unrecognized switch
Multiple conflicting switches
File specification syntax error
No source input (neither file nor source statements)

4.12.3.12 $SORT[1] - The $SORT control statement runs the PDP-11 Sort program SORT11 (not the RSTS-11 program SORT.BAC). The SORT11 program is on RSTS/E systems with the COBOL compiler. For more information on the PDP-11 program SORT11, refer to the PDP-11 SORT Reference Manual.

The format of the $SORT control statement is as follows.

```
$SORT[switches] [file specification[switches]]
```

The following are the valid switches for the command field.

/SIZE:n	maximum record size in bytes. Can be between 1 and 16383.
/FILES:n	number of scratch files to be used. Can be between 3 and 10.
/PROCESS:x	use the sort process given by x. Values can be R (record sort), T (tag sort), A (ADDROUT sort), or I (index sort). If /PROCESS:x does not appear or if :x is omitted from the /PROCESS:x switch, a record sort is performed.

[1]This feature is not available prior to Version 6A (RSTS/E) systems.

/KEYS:abm.n	use the sorting key field defined by entries for a, b, m and n. Maximum of 10 keys can be specified if each is separated by a colon as follows.
	/KEYS:abm.n:abm.n: ... abm.n:abm.n
/RIN	specifies the input file as a COBOL relative file. Without /RIN, input file is assumed to be a sequential ASCII file.
/ROUT	specifies the output file as a COBOL relative file. Without /ROUT, the output file is created as a sequential ASCII file.

The command field of the $SORT control statement must have the /SIZE:n switch to define the record size. The requirements for other command field switches depend on file specifications present and the type of sort requested.

A maximum of three file specifications can appear in the $SORT control statement: an input file, an output file, and a specification file. To distinguish these files, the following switches are used.

/INPUT	the file to be sorted
/OUTPUT	the file to contain the sorted data
/SPECIFICATION	the file which contains the control information for the sorting process.

A file specification without a switch is used as the file to be sorted. If the /SPECIFICATION switch is used, the /KEYS and /PROCESS switches must not appear in the command field. If a specification file is not given in the control statement, the /KEYS switch must be included in the command field to control the sorting process.

Missing elements in a file specification are replaced by BATCH default elements. If type is omitted from the file specification, BATCH uses SRT as the type.

The following is a sample batch stream using $SORT commands.

```
$JOB/NAME=SRTØØ2/LIMIT=3Ø
$SORT/PRO:T/SIZ:1ØØ/KEY:O4.1:O1.1 F1ØØ.1ØØ A/OUT
$SORT/PRO:T/RIN/KEY:1.4/SIZ:1ØØ R2ØØ.1ØØ/INP B/OUT
$SORT/SIZ:1ØØ/ROU/KEY:1.4/PRO:T F1ØØ.1ØØ/INP X/OUT
$SORT/SIZ:6Ø/KEY:3.5/FIL:3 V1ØØ.Ø6Ø/INP C/OUT
$SORT/SIZ:1ØØ/RIN R2ØØ.1ØØ/INP D/OUT SPEC.ØØ1/SPE
```

The /SIZE switch appears in each command to define the record size in the file to be sorted. In the first $SORT control statement, file F1ØØ.1ØØ is used as the input file. In the second statement, the /RIN switch in the command field denotes the input file R2ØØ.1ØØ as a COBOL relative file. In the third statement, the /ROUT switch causes the output file X.SRT to be a

COBOL relative file. For the fourth $SORT control statement, a record sort is performed on file V1ØØ.Ø6Ø because the /PROCESS:x switch is absent. For the fifth $SORT statement, the sorting process is controlled by data in the specification file SPEC.ØØ1. The /PROCESS:x and /KEYS switches are not permitted. For all files without a type in the specification, BATCH uses the default SRT.

4.12.4 Batch Operating Procedures

This section describes how the RSTS/E system user requests batch processing and how the batch processor generates output.

4.12.4.1 Requesting a Batch Job Run - To request the running of a batch job, the user runs the library program QUE and specifies the batch control file or files as follows.

```
RUN $QUE
QUE      V06A-02 - RSTS V06A-01 SYSTEM #880
#Q BA:BATJOB=FILE1,FILE2,FILE3.DAT
#
```

The user normally queues a batch job to device BA:. The job and log files in this example will be named BATJOB, and the files FILE1.CTL, FILE2.CTL, and FILE3.DAT will be concatenated to form the batch control file. The log file BATJOB.LOG will be printed after the job is complete.

4.12.4.2 Batch Processing - As the batch control file is read, it is checked for command sequence and syntactical validity. If an error is detected, an error message is printed in the log file. The job will not be run, but syntax checking will continue through the remainder of the file.

If no errors are detected, the job is processed. A log is printed, showing the sequence of Batch commands processed during the course of the job. If program output is directed to KB:, this output appears following the command that caused the program to execute. In the example that follows, a BATCH job named JOB1 has been run. The Batch control file contained the following sequence of commands:

```
$JOB/NAME=JOB1/LIMIT=4
$CREATE SUB1.BAS
        source statements
$EOD
$BASIC/RUN LISTING/LIS MAIN/OBJ
        source statements
$DATA
        data
$PRINT SUB1.BAS
$EOJ
```

These commands have the following effect:

```
$JOB/NAME=JOB1/LIMIT=4
```

A job name of JOB1 is assigned to the job. This name appears on the job log along with the time and date of the job's execution. A time limit of four minutes is set. If the job is not finished in four minutes from its start (actual elapsed time), the job is terminated, and the appropriate error message is printed in the log.

```
$CREATE SUB1.BAS
```

A BASIC source file named SUB1 is created, from data records which must follow the $CREATE command.

```
$EOD
```

The $EOD command signals the end of SUB1.BAS.

```
$BASIC/RUN LISTING/LIS MAIN/OBJ
```

The source file following this command is compiled; a listing of the source statements is created in a file named LISTIN.LIS; and the resulting object file (MAIN.BAC) is saved and is executed.

```
$DATA
```

The data to be read during execution of MAIN.BAC follows this command.

```
$PRINT SUB1.BAS
```

The source file created by $CREATE SUB1.BAS is printed. This command also has the effect of terminating data input to MAIN.BAC.

```
$EOJ
```

This command signals the end of job JOB1.

4.12.4.3 Error Procedures - When an error is detected in a batch command, the job is not started. Instead, an error log is printed listing all commands and data scanned along with the appropriate error message(s). The batch log file always indicates all command lines scanned. If an error is identified on a command line, the error message follows the command, marked with question marks (??????????). Scanning of the control file continues, but the job will not be executed.

If no errors occur, the time of output of lines will be indicated in the left margin of the log. All normal terminal interaction corresponding to the Batch commands will appear in the log.

THIS PAGE PURPOSELY LEFT BLANK.

4.13 DIRECT PROGRAM - DIRECTORY LISTINGS[1]

The DIRECT program lists file related information from a disk directory. The benefits of DIRECT are increased speed and more options compared with other methods of listing directories. DIRECT opens a user's directory as a file and reads information by immediately accessing the blocks in the directory. This action is faster than the conventional method of passing the request for such information to certain system functions. Since the program follows linked pointers through a disk directory, the pointers must not be changed during program execution.

The user runs DIRECT by typing the RUN $DIRECT command or by using the CCL command explained at the beginning of this chapter. When DIRECT runs as a result of the RUN command, it prints the # character which acts as a prompt indicator. The user can then type a command to DIRECT. If the user runs DIRECT by the CCL command, he includes the command to DIRECT in the CCL command.

The general format of the command is as follows:

output=input/option(s), input/option(s),...

Output is optional and can be a device specification or a disk file specification. If output is not given, the = character is optional and DIRECT prints output at the user's terminal. If an extension does not appear with an output filename, DIRECT appends .DIR unless the user forces a null extension by specifying the . character with the filename. Input can be any number of full disk file specifications. The full disk file specification on input can include a device, filename, extension and project-programmer number. If a device is not given, DIRECT uses the public structure and denotes this by the SY: specification. The filename, extension, and project-programmer fields can contain * and ? characters to denote wild card specifications. If no file specification is given or if an * character is given as the file specification, DIRECT processes all files in the directory. DIRECT applies the default interpretations shown for the following specifications for a given directory.

null	*.*	All files
*	*.*	All files
*.	*.	All files with null extensions
.	*.	All files with null extensions
.EXT	*.EXT	All files with extension EXT
FILE	FILE.*	All files with filename FILE

[1]This feature is not available before Version 5 (RSTS/E) systems.

With a file specification, the user can specify a /character followed by one or more options. If no options appear, DIRECT proceeds as if the user had specified /DI. Table 4-24 lists and describes the options.

Table 4-24

DIRECT Options

Type	Format	Meaning
Individual	/NA	List filenames only.
	/EX	List filenames and extensions of each file.
	/SI	List filename, extension and list size of each file as number of 256 word blocks occupied.
	/PR	List filename, extension and file protection code.
	/LA	List filename, extension and date of last access for each file.
	/DA	List filename, extension and date of creation for the file.
	/TI	List filename, extension, date and time of day when file was created.
	/CL	List filename, extension, and file cluster size.
	/SU	List only summary data to include number of designated files and number of blocks occupied by designated files.
Aggregate	/BR /F	List filenames and extensions with a brief summary message.
	/DI:S /S	List all relevant data to include headings, filenames, extensions, size, protection code, date of last access, date of creation, time of creation, clustersize and summary. (Called slow directory.)
	/DI null	List most important data to include heading, filename, extension, size protection code, date of last access and summary.

Table 4-24 (Cont.)

DIRECT Options

Type	Format	Meaning
General	/HD	Print heading at top of columns on the listing.
	/W	List data across the width of a line rather than one item per line. Useful with large directory listings and individual options.
	/HE	Print the file DIRECT.HLP which describes the DIRECT program.
	/BK	List the directory for the specified device in reverse chronological order. As a result, the most recently created files appear at the beginning of the listing. If /BK is used to list files in the public structure and multiple disks are in the public structure, the listing reflects reverse order for each disk.

To list a directory at the line printer, the user specifies the device designator and options in the command. For example,

```
# LPØ:=*.BA?/F/W
#
```

The command lists filenames and extensions of all files with BA in the extension. DIRECT formats the data across the width of the line printer paper.

To list directories of several accounts and place them in a disk file, specify the project-programmer field and the disk file specification in the command. For example,

```
# PROJ=[120,*]/DI
#
```

DIRECT creates the file PROJ.DIR in the current account and writes, to the file, directory listings of all accounts with project number 120.

An error encountered in a command causes DIRECT to print a message followed by the # character. The user must retype the entire command correctly. The messages are described in Table 4-25.

4.13.1 DIR as a CCL Command

The following commands show some useful methods of requesting listings with the standard CCL command DIR. To list the filenames and extensions of all files in the user's account, type the following command.

```
DIR /BR/W
```

DIRECT prints the listing at the user's terminal and lists the information across the width of the page. To list at another device the filenames, extensions, sizes, protection codes and creation dates for certain files in the current account, type the following command.

```
DIR LP1:=*.BAS
READY
```

DIRECT prints the listing of all BASIC-PLUS source files on line printer unit 1. The READY message indicates DIRECT has completed printing. To obtain a reverse listing, type the following command.

```
DIR /DI/BK
```

DIRECT prints at the current terminal, directory and summary information in reverse chronological order. To print the file DIRECT.HLP on the line printer, type the following command.

```
DIR LP1:=/HE
READY
```

Table 4-25

DIRECT Program Error Messages

DEVICE NOT DIRECTORY STRUCTURED	Device specified does not use a directory for file access.
DIRECTORY OF dev:[n,m] IS EMPTY	DIRECT finds of the account [n,m] on device dev: contains no entries.
DISK PACK IS NOT ON-LINE	The pack or cartridge referred to is either not mounted or is off-line.
ILLEGAL FILE NAME <filename>	The file specified by <filename> contains a logical device name which the user has not reserved by the ASSIGN command.
ILLEGAL INPUT FILE SPEC file spec	The file specification indicated by file spec generates RSTS error number 2, ILLEGAL FILE NAME.
ILLEGAL SWITCH text	File specification contains an undefined option indicated by text.
INVALID DEVICE SPECIFICATION	The device specification is invalid or the device referred to does not exist on the system.
NO DIRECTORY FOR [n,m] ON dev:	DIRECT cannot find an account for user account [n,m] on the device dev: or else DIRECT encounters a protection violation.
NO HELP AVAILABLE	The file DIRECT.HLP is not on the system library account.
NO SUCH FILE AS <file spec> ON [n,m].	DIRECT cannot find the requested file indicated by <file spec> in the account [n,m].
OUTPUT FILE MUST BE IN THE USER'S AREA	DIRECT does not create an output disk file in another account if user is not privileged.
TOO MANY FILES FOR INVERTED DIRECTORY LISTING	DIRECT limits use of the /BK option to accounts with less than 200 files.

THIS PAGE PURPOSELY LEFT BLANK

4.14 UMOUNT PROGRAM - MOUNTING AND DISMOUNTING PRIVATE DISKS[1]

A non-privileged or privileged user can execute the MOUNT or DISMOUNT commands of the UMOUNT system program to logically mount and dismount private disk packs and cartridges. The commands are executed only if the standard CCL feature is available on the system. Otherwise, the system prints the WHAT? error message. An attempt to run the UMOUNT program by any other means generates the message PLEASE USE THE 'MOUNT' OR 'DISMOUNT' COMMAND.

To logically mount a private pack or cartridge on the system, perform the following procedure.

a. Load the pack or cartridge in the available drive. Ensure that the device is available by running the SYSTAT system program.

b. Ensure that the drive is write enabled.

c. Type the MOUNT command followed by the device designator and the appropriate identification label.

For example,

```
MOUNT DK1:MYPACK
READY
```

The UMOUNT program runs and logically associates RK11 drive unit 1 with the identification label MYPACK. The READY message indicates that the disk is available for read and write access and in the UNLOCK state.

An attempt to write a file on the disk when the current account is nonexistent generates the DISK PACK IS PRIVATE error (ERR = 24). The user must request the system manager or responsible system programmer to create the current account on the disk.

If any error occurs in the device designator or in the identification label, the program prints the following error message.

[1] This feature is not available prior to Version 5B (RSTS/E) systems.

ERROR IN MOUNT - <text>

The text indicates the related RSTS-11 error encountered. The program terminates.

If any error occurs in mounting the disk on the system, the program prints the following error message:

PROCESS ERROR IN MOUNT - <text>

The <text> indicates the related RSTS-11 error encountered. Some typical errors include having the device protected against writing (DEVICE HUNG OR WRITE LOCKED), specifying an incorrect identification label (PACK IDS DON'T MATCH) and trying to mount a disk which is already mounted (DISK IS ALREADY MOUNTED).

If the program encounters an error when attempting to unlock the disk to enable write access, it prints the following message:

ERROR UNLOCKING MOUNTED PACK - <text>

The text indicates the related RSTS-11 error encountered. However, the pack is mounted and available for read access only. The user should report any error of this kind to the system manager or responsible system programmer.

The UMOUNT program accepts options in the MOUNT and DISMOUNT commands[1]. If the disk to be mounted must be kept locked, simply append the /LOCK option to the MOUNT command. For example:

```
MOUNT DB1:DATA1/LOCK
READY
```

UMOUNT runs and logically associates the identification label DATA1 with RH11/RP04 unit 1. The READY message indicates that the disk pack is in the LOCK state. Only privileged users have read and write access to the disk pack

The MOUNT command can be used with a magtape designator to assign a unit to the current job and to set default labeling format. The following example shows the procedure.

```
MOUNT MT1:/DOS
READY
```

[1] This feature is not available before Version 6A systems.

Magtape unit 1 is assigned to the current job with DOS/BATCH labeling default. The /ANSI option sets the labeling default to ANSI standard format.

If the current job is privileged, the /JOB:n option can be used with a magtape designator and the MOUNT command to reassign the unit to job number n. For example,

```
MOUNT MT1:/JOB:5
READY
```

Magtape unit 1 is reserved to job 5 with the currently assigned default labeling format. If job 5 is not active, an error message is printed. If an unassigned logical name is given for the magtape unit in the MOUNT command, UMOUNT prints the error text ERROR IN MOUNT - ILLEGAL DEVICE and returns to the READY level. If a name is given with the magtape designator in the MOUNT command, UMOUNT prints the message ERROR IN MOUNT - SYNTAX ERROR and returns to the READY level.

To logically dismount a private pack or cartridge, perform the following procedure.

a. By use of SYSTAT, determine the number of OPEN files on the device. If non-zero, wait until all files are closed before proceeding. If zero, proceed.

b. Type the DISMOUNT command followed by the device designator of the drive.

For example,

```
DISMOUNT DK1:
READY
```

The UMOUNT program runs and logically dismounts the pack on RK11 drive 1. The pack identification is removed from the monitor's table of logical names. The READY message indicates that the drive is free for other usage and that the user can safely remove the pack or cartridge from the drive.

If the program encounters an error when it attempts the dismount action, it prints a message in the following format.

```
PROCESS ERROR IN DISMOUNT - <text>
```

The <text> indicates the related RSTS-11 error encountered. A typical error is attempting to dismount a disk which has open files (ACCOUNT OR DEVICE IN USE).

To rewind a magtape and take it off line, use the /UNLOAD switch in the DISMOUNT command. For example:

```
DISMOUNT MTØ:/UNLOAD
```

The magtape on unit 0 is rewound and placed off line.

4.15 FILCOM PROGRAM[1]

The FILCOM (file compare) system program compares two ASCII files, line by line, and prints the differences found. The user must specify which two files are to be compared as well as how many successive lines must be compared at a time. This second consideration is especially important for isolating differences in programs' subroutines.

FILCOM is called as follows:

```
RUN $FILCOM
```

The first query line printed is:

```
OUTPUT TO?
```

Type the device designators of the peripheral device on which the comparison information is to be printed. Typing the CR key alone specifies the keyboard on which the user is working.

The next two printed query lines request the names of the files to be compared. Be sure to use the full name of each file, including its extension. For example:

```
INPUT FILE #1? SORT1.BAS
INPUT FILE #2? SORT2.BAS
```

Any number of lines can be compared at a time. The number of successive lines that determines a match is specified by the user response to the FILCOM query line as follows:

```
HOW MANY TO MATCH?
```

Responses to this query are discussed in detail later in this section. If the CR key is typed alone, the number of successive lines to match is automatically assumed to be three.

The next query line is:

```
BASIC+ LINES?
```

Type the letter Y in response to this query to instruct the program to consider BASIC-PLUS continuation lines as part of the numbered line. In this way, multiple line statements can be compared. For example, consider the line shown below.

[1]This feature is not available prior to Version 5B (RSTS/E) systems.

```
100 FOR J=10 TO 15:
        PRINT 3.14*J/2:
    NEXT J
```

If Y were typed in response to the BASIC+ LINES query, FILCOM would compare all three statements (FOR, PRINT and NEXT) as a single line. By responding with any other character (or no character), FILCOM would compare only the statement printed on the first line (i.e., FOR J=10 TO 15) to a single line in the other file. Typing Y ensures that FILCOM prints the line numbers of any BASIC-PLUS lines found to be different.

The next query line is:

BLANK LINES?

Typing the letter Y in response to this question instructs the program to compare blank lines. If the Y is not specified, FILCOM skips all blank lines.

Assume the following two files are on the user's disk:

```
TEST1
10 REM A
20 REM B
30 REM C
40 REM D
50 REM E
60 REM F
70 REM G
80 REM H
90 REM I
100 REM J
110 REM K
120 REM L
130 REM M
140 REM N
150 REM O
160 REM P
170 REM Q
180 REM R
190 REM S
200 REM T
210 REM U
220 REM V
230 REM W
240 REM X
250 REM Y
260 REM Z
```

```
TEST2
10 REM A
20 REM B
30 REM C
70 REM G
80 REM H
90 REM I
100 REM J
110 REM 1
120 REM 2
130 REM 3
140 REM N
150 REM O
160 REM P
170 REM Q
180 REM R
190 REM S
200 REM T
210 REM U
220 REM V
222 REM 4
224 REM 5
230 REM W
240 REM X
250 REM Y
260 REM Z
```

EXAMPLE 1

To compare these two files, TEST1 and TEST2, FILCOM is run as follows:

```
RUN $FILCOM
OUTPUT TO?
INPUT FILE #1? TEST1.BAS
INPUT FILE #2? TEST2.BAS
HOW MANY TO MATCH?
BASIC+ LINES?
BLANK LINES?

********************
1) TEST1.BAS
40 REM D
50 REM E
60 REM F
70 REM G
*****
2) TEST2.BAS
70 REM G
********************
1) TEST1.BAS
110 REM K
120 REM L
130 REM M
140 REM N
*****
2) TEST2.BAS
110 REM 1
120 REM 2
130 REM 3
140 REM N
********************
1) TEST1.BAS
230 REM W
*****
2) TEST2.BAS
222 REM 4
224 REM 5
230 REM W

 3 DIFFERENCES FOUND


READY
```

Notice that FILCOM prints the lines that do not compare equal in two files followed by the first line that does compare equal. The first line in each group is the first line that does not compare and the last line in each group is the first line that does compare. In general, the program does not print groups of lines that are identical. FILCOM first prints line 40 of TEST1, since it is the first line that does not appear in TEST2. This line is considered the first difference. FILCOM continues to print TEST1 lines until it locates 3 lines that can be matched against 3 lines in TEST2. In this case, lines 70, 80, and 90 of both TEST1 and TEST2 are identical, so the printout for this difference is ended.

Since lines 80 through 100 are identical in both files, FILCOM does not print them. The next lines that do not compare are lines 110 through 130. These lines are printed under both files. This is considered the second difference. FILCOM continues to print lines until it locates 3 matching lines. Lines 140 through 160 are identical for both files.

Since lines 150 through 220 are identical in both files, FILCOM does not print them. The next lines that do not compare are 222 and 224 of TEST2. Consequently, they are printed under the TEST2 heading. This is considered the third difference. FILCOM continues to print TEST2's lines until it locates 3 lines that can be matched against 3 lines in TEST1. Lines 230 through 250 are identical for both files.

Finally, FILCOM prints the number of differences found. If one of the two files compared were empty (e.g., an empty data file), FILCOM would have printed, A NULL FILE??

EXAMPLE 2

Files TEST1 and TEST2 can be compared in another way. To compare these files for eight consecutive identical lines instead of three, FILCOM is run as follows:

```
RUN $FILCOM
OUTPUT TO?
INPUT FILE #1? TEST1.BAS
INPUT FILE #2? TEST2.BAS
HOW MANY TO MATCH? 8
BASIC+ LINES?
BLANK LINES?

********************
1) TEST1.BAS
40 REM D
50 REM E
60 REM F
70 REM G
80 REM H
90 REM I
100 REM J
110 REM K
120 REM L
130 REM M
140 REM N
*****
2) TEST2.BAS
70 REM G
80 REM H
90 REM I
100 REM J
110 REM 1
120 REM 2
130 REM 3
140 REM N
********************
```

```
1) TEST1.BAS
230 REM W
*****
2) TEST2.BAS
222 REM 4
224 REM 5
230 REM W

 2 DIFFERENCES FOUND

READY
```

In this case, the first difference found, again, is line 40 in TEST1. Instead of matching lines 70 of both files, FILCOM begins to match line 140, since 140 to 220 is the next group of at least eight successive identical lines. Although lines 70 through 100 are identical in both files, they form less than eight successive lines.

Now assume the following two files are on the user's disk:

FILE 1

```
THIS IS LINE 1
THIS IS LINE 2

THIS IS THE LINE AFTER THE BLANK LINE
THIS IS THE NEXT LINE
```

FILE 2

```
THIS IS LINE 1
THIS IS LINE 2

THIS IS THE LINE AFTER THE BLANK LINE
THIS IS THE NEXT LINE
```

Notice that file 2 has an extra blank line.

EXAMPLE 3

To compare these two files, FILCOM is run as follows:

```
RUNNH
OUTPUT TO?
INPUT FILE #1? FILE1
INPUT FILE #2? FILE2
HOW MANY TO MATCH?
BASIC+ LINES? NO
BLANK LINES? NO


0 DIFFERENCES FOUND
```

Since no response was typed after the query BLANK LINES?, FILCOM did not compare blank lines. Typing the letter Y in response to the BLANK LINES query, however, results in the printout shown below.

```
RUN $FILCOM
OUTPUT TO?
INPUT FILE #1?FILE1
INPUT FILE #2?FILE2
HOW MANY TO MATCH?
BASIC+ LINES? NO
BLANK LINES? YES

********** ********
1) FILE1
THIS IS THE LINE AFTER THE BLANK LINE
*****
2) FILE2

THIS IS THE LINE AFTER THE BLANK LINE

 1 DIFFERENCES FOUND


READY
```

If the user includes /P with a file name in response to the OUTPUT TO query, FILCOM creates the output file as a patch file. With the APPEND command, such a patch file can later be included in the file specified as INPUT FILE#2. This action makes the resultant file equivalent to the BASIC-PLUS program specified as INPUT FILE #1. To create the patch file, the user must also answer both the BASIC+ LINES and BLANK LINES query with YES.

4.16 INUSE PROGRAM[1]

The INUSE system program prints or displays the words, "IN USE" in block letters on the terminal to warn others not to use it. This message is followed by the user's job number and account number.

INUSE is called as follows:

```
RUN $INUSE
```

The printout from this program is shown below.

```
RUN $INUSE
  IIIIII     NN       NN              UU       UU    SSSSSS     EEEEEEEEEE
  IIIIII     NN       NN              UU       UU    SSSSSS     EEEEEEEEEE
    II       NNNN     NN              UU       UU  SS       SS  EE
    II       NNNN     NN              UU       UU  SS       SS  EE
    II       NNNN     NN              UU       UU  SS           EE
    II       NNNN     NN              UU       UU  SS           EE
    II       NN  NN   NN              UU       UU    SSSSSS     EEEEEE
    II       NN  NN   NN              UU       UU    SSSSSS     EEEEEE
    II       NN    NNNN               UU       UU           SS  EE
    II       NN    NNNN               UU       UU           SS  EE
    II       NN    NNNN               UU       UU  SS       SS  EE
    II       NN    NNNN               UU       UU  SS       SS  EE
  IIIIII     NN      NN                 UUUUUU       SSSSSS     EEEEEEEEEE
  IIIIII     NN      NN                 UUUUUU       SSSSSS     EEEEEEEEEE

        BY JOB 16 USER [120,80]
```

Type the CTRL/Z or CTRL/C combination or execute any command to regain control of the terminal at any time.

[1]This feature is not available prior to Version 5B (RSTS/E) systems.

THIS PAGE PURPOSELY LEFT BLANK

4.17 COPY PROGRAM[1]

All of the information on a DECtape, magtape or disk can be copied by using the COPY system program and the /FC (fast copy) option. COPY is called as follows:

```
RUN $COPY
```

COPY prints a pound sign (#) when it is ready to receive instructions. To print a help message, simply type /HE. To copy a device, the user must respond in the format shown below.

```
#new device<old device /FC
```

For example:

```
#DT1:<DT2:/FC
```

In this example, all of the data, programs and directories are copied, block by block, from the DECtape on unit 2 to the DECtape on unit 1. The original information on unit 2 is, of course, still intact.

Notice that all of the information on a device is copied; individual files cannot be specified.

Magtapes, DECtapes, disk cartridges and disk packs can be copied. In addition, information can be copied only to different units of the same device. That is, COPY can be used to copy a DECtape to another DECtape or a magtape to another magtape, but not to copy, say, a DECtape to magtape or vice versa. If an attempt is made to copy the information on one type of device onto another type of device, the error message MUST HAVE SAME TYPE DEVICES is given.

Finally, if the same unit number of a device is specified twice, the error message MUST HAVE DIFFERENT DEVICE UNITS is given.

When the information is successfully copied from one device to another, the program terminates and the system prints READY.

[1] This program is not available prior to Version 5B (RSTS/E) systems.

To verify that the information on a device unit has been copied properly, use the /VE (verify) option with the COPY program in either of the following ways:

```
#DT1:<DT2:/FC/VE
```

or

```
#DT1:<DT2:/NC/VE
```

Since verification is performed block by block, it requires as much time as the copy operation. Therefore, it is important to understand the difference between the above commands. The first command shown above, which includes the /FC option, copies information from DT2 to DT1 and then verifies that the information was copied correctly. The second command, which includes the /NC (no copy) option, does not copy information; it only verifies that the information on both DECtapes is identical.

The two sample commands shown above are the only allowable forms for verifying. If the user types a command string omitting the /FC or /NC option, the error message /FC OR /NC MISSING -- PLEASE TRY AGAIN is returned. Typing a command string without specifying an option, gives the error message NO USEFUL OPTION SPECIFIED -- PLEASE TRY AGAIN.

As with the /FC option, individual files cannot be specified with the /VE option; all of the information on a device is verified.

As the verification is performed, the first message COPY prints is:

```
BEGINNING VERIFICATION PASS
```

If all of the information has been copied correctly from one device to another, the next message COPY prints is:

```
VERIFICATION COMPLETE Ø BAD BLOCKS
READY
```

If, however, the information has not been copied correctly, COPY prints the decimal number of the blocks in which inconsistencies appear. For example:

```
#DT1:<DT2:/FC/VE
BEGINNING VERIFICATION PASS
THE FOLLOWING BLOCKS ARE BAD:
17
31
89

VERIFICATION COMPLETE 3 BAD BLOCKS


READY
```

The /BL option, which specifies blocksize, speeds up copying or copies magtapes written with non-standard record sizes. To speed up copying, specify a larger blocksize. For example:

```
#DKØ: <DK1:/BL:2Ø48/FC
```

causes the disk on drive DK1: to be copied to the disk on drive DKØ: in 2048-byte blocks, rather than the default 512-byte blocks.

To specify other than standard density and parity settings when copying magtape, use the /DENSITY:d and /PARITY:p switches where

d can be:	p can be
2ØØ	ODD
556	EVEN
8ØØ	
DUMP	
16ØØ	

(16ØØ implies phase-encoded.) For example,

```
#MTØ:/DENSITY:556/PARITY:EVEN <MT1:/DENSITY:DUMP/FC
```

COPY reads magtape unit 1 at 800 bits per inch dump mode and writes unit 0 in even parity at 556 bpi.

THIS PAGE PURPOSELY LEFT BLANK.

4.18 EXTENDED PIP PROGRAM[1]

The PIP system program has been improved to provide the following features:

a. accepting wild card characters in file name and extension specifications,

b. inspecting eligible files for transfer, rename, and delete operations,

c. pre-extending or extending an output file,

d. writing zeroes over data before a file is deleted,

e. reading DOS/BATCH format disks for copy and directory listing operations,

f. handling ANSI format magtapes on transfer and directory listing operations,

g. processing indirect commands, and

h. continuing the command on the next physical line.

The improved PIP is a new program, referred to as extended PIP. It requires a 16K word job area to run and is therefore optional on all systems. The standard 8K word version of PIP has not been altered. Extended PIP replaces the standard version on systems having the QUEMAN and SPOOL programs. To confirm the version of PIP available, simply type /HE while at PIP command level and the help file will be printed.

Extended PIP has all the features of the standard version and includes those new features described in the following sections. For general information on PIP, refer to Section 4.4.

NOTE

For wild card processing in delete and rename commands, extended PIP uses a temporary file under the user's account to store directory information. If the user's account is full, PIP encounters the NO ROOM FOR USER ON DEVICE error when it attempts to open the temporary file. To overcome the error, use the KILL system command to delete one file and then rerun PIP.

[1]This feature is not available prior to Version 6A (RSTS/E) systems.

4.18.1 Wild Card Specifications

Extended PIP allows * and ? characters in file names and extensions to denote wild card elements and to thereby designate multiple files in a single file specification.

The asterisk character (*) replacing a file name, an extension, or both denotes all file names, all extensions, or all file names with any extensions. The following samples demonstrate the usage of the * character.

FILE.*	All files with file name FILE and any extension
*.EXT	All files with EXT extensions
.	All files

The question mark character (?) in any position of either the filename or extension means that any alphanumeric character can be substituted in that position to designate a file. The following samples demonstrate the usage of the ? character.

FILE.EX?	All files with name FILE and EX as the first two characters and any other character as the third character in the extension.
FILE??.EXT	All files with FILE as the first 4 characters and any characters as the fifth and sixth characters of the name and with EXT as the extension.
FILE??.E??	All files with FILE as the first 4 characters and any characters as the fifth and sixth characters of the name and with E as the first character and any characters as the second and third characters of the extension.
FI?.EXT	All files with an extension of EXT and with 3-character names, of which the first 2 characters are FI and the third character is any character.

The * and ? characters can be intermixed in a file specification. The following samples show the default interpretations.

FILE??.*	All files with any extension and with FILE as the first 4 characters and any characters as the fifth and sixth characters of the name.
*.EX?	All files with EX as the first 2 characters and any character as the third character of the extension.

In the absence of an element or elements of the full file specification, extended PIP substitutes defaults for file names or extensions or both. The following default interpretations are applied to the incomplete specifications shown.

Spec	Default	Meaning
null	*.*	All files
*	*.*	All files
*.	*.	All files with null extensions
.	*.	All files with null extensions
.EXT	*.EXT	All files with extension EXT
FILE	FILE.*	All files with name FILE

4.18.2 Extended PIP Defaults and Additional Options

Wild card characters allow multiple files to be designated in the input and output of a PIP command. Multiple file specifications can be given on input: each must be separated by a comma. Where multiple file specifications appear as input in a PIP command, the default interpretations shown in Table 4-26 apply for missing elements in each specification.

Table 4-26

General Input Defaults

Missing Element	Default Interpretation
Account	The previously specified account. If missing from the first file specification given, the current user's account is the default.
Device	The previously specified device. If missing from the first file specification given, the public structure (SY:) is the default.
File name	The asterisk (*) specification
Dot and extension[1]	The asterisk (*) specification

[1] If the dot is present, the extension is assumed to be present but can be null. When the dot is not present (and only then), the default extension is used.

On output, however, only one file specification is permitted although that file specification can designate multiple files. Only the asterisk (*) is permitted as a wild card character in an output specification. An attempt to use the question mark (?) in the output generates the INVALID DEVICE error. When the output is a wild card specification, the input specification is repeated for each file created as output.

Extended PIP applies specific default values for all elements in a transfer or directory listing command. The input and output defaults are summarized in Tables 4-27 and 4-28. The defaults for rename and delete commands differ from those applied to transfer commands. (Sections 4.18.4 and 4.18.5 describe the individual differences.)

Table 4-27

Output Defaults for Transfer and Directory Commands

Element	Description	Default
dev:	device specification	The public structure (SY:)
[proj,prog]	project and programmer (account) number	Current user account number
name	file name specification	Input file name is used
.ext	file name extension	The input specification is used
<prot>	protection code	The system wide default (<60>), the default set by the ASSIGN command, or the value specified in the input.
null	no specification	The current keyboard (KB:)

Table 4-28

Input Defaults for Transfer and Directory Operations

Element	Description	Default
dev:	device specification	Immediately previous device specification. If none, then SY: is substituted.
[proj,prog]	project and programmer (account) number	Immediately previous account number. If none, then current user account number is substituted.
name	file name specification	All files (Extended PIP substitutes the * character)
.	dot	All files with null extensions
.ext	file name extension	All extensions (Extended PIP substitutes the * character)
<prot>	protection code	The system wide default or the default set by the ASSIGN command.
null	no specification	All files on the immediately previous device and account. If none, SY: and current user account are used.

The new options available in extended PIP are described in Table 4-29. Either the < or = character can separate the input and output sides of the PIP command.

Table 4-29

Additional Extended PIP Options

Option	Function
/IN	Used in transfer, delete, and rename operations to inspect each file eligible for the operation. PIP prints the full specification of an eligible file and the ? character. To execute the operation for that file, YES (or any string beginning with Y) must be typed. To omit the operation for that file, NO (or any string not beginning with Y) can be typed. The /IN option affects the individual file specification and not the entire command. For example, #A.*,B.*/IN.C.*/DE PIP deletes the files with names A and performs the inspect action for files with name B. It then deletes files with name C.
/PRX:n	Used on transfer operations to pre-extend a disk file to n blocks. If :n is omitted and the first input file is from disk or DECtape, pre-extend the output file to the length of the first input file. Cannot be used with the /UP option.
/EX	Used on transfer operations to open the disk output file with MODE 2 (for appending date). (See Section 10.5.2 of the BASIC-PLUS Language Manual.) The output file is extended by appending the input file(s) to it. Cannot be used with the /PRX, /CL, and /UP options.
/WO	Used on individual disk file specifications in delete commands. PIP writes zeroes in the file before it is deleted. The /WO option must appear with each file specification to which it applies, whereas the /DE option need appear only once in the command string. For example, #A.BAK/WO,B.BAK/DE The /DE option applies to both files but the /WO option applies only to the file A.BAK.
/VID:label	Required when using the /ZE option on ANSI standard magtape. The label is an alphanumeric volume label of the magtape to be zeroed. The following example shows the format of the command. #MT1:/VID:ABC/ZE

Table 4-29 (Cont.)

Additional Extended PIP Options

Option	Function
/VID:label (cont.)	The reel mounted on unit 1 is read. If the reel is in ANSI format, PIP prints the REALLY ZERO query only if the /VID: option is present and has a valid identification. A response of Y zeroes the magtape using the label ABC. If the reel is not ANSI format, PIP prints the REALLY ZERO query to allow the tape to be zeroed with the volume label ABC. (See Section A.3 of the RSTS/E Programming Manual for a description of a zeroed magtape.)
/HE	Prints the help file PIP.TXT. /HE must be the only element in the input specification.
/LI	Creates a directory listing. For RSTS/E disk, DECtape, and DOS format disk, the following data is printed: name, extension, length (C denotes contiguous for DOS disk files), RSTS/E protection code, and creation date. For DOS label magtape, the DOS protection code replaces the RSTS/E protection code and both the RSTS/E project-programmer number and the DOS user identification code are added to the listing. For ANSI label magtape, the protection code is not applicable and is omitted.
/LI:S	Creates a full (slow) directory listing. For RSTS/E disks only. Adds time of creation, date of last access, and file cluster size information to the standard directory listing. For other devices, prints the standard directory listing.
/DOS	Indicates that the input disk is in DOS/BATCH format. PIP can read the disk but cannot write on the disk. Thus, only transfer and directory operations are possible.
/BLOCK	Force an ANSI magtape input file to be read as if it were written in ANSI:U format.
/FORMAT:U	Forces an ANSI magtape output file to have an undefined format (contiguous mode transfer required). If /BL:n is on the input file specification, the output file block length is n bytes.
/FORMAT:V	Forces an ANSI magtape output file to have a variable length record format. If /BL:n is on the input file specification, the output file block length is n bytes.
/MORE	Continues a command on the next physical line.

4.18.3 Wild Card Specifications in Transfer and Directory Listing Commands

For transfer and directory listing requests in extended PIP, the standard defaults described in Tables 4-27 and 4-28 are applied. The sample commands and accompanying text in this section demonstrate the defaults applied to transfer requests. The same defaults also apply to directory listing requests. The following example illustrates the defaults applied to a multiple file transfer request.

```
#DK1:*.BAK<*.BAS
```

PIP transfers all files with BAS extensions from the current account in the public structure. Each eligible file is created as a unique file on RK unit 1 under the current account. Each file has the same file name as the input file but is given an extension of BAK and the default protection code.

The following command shows the device and account defaults used when elements are missing from the input.

```
#DK1:<6Ø><MTØ:[2,2]*.MAC,*.BAK
```

PIP first transfers all files with MAC extensions from account 2,2 on magtape unit Ø to the current user account on RK unit 1. In the absence of a device and account designation in the second input specification, PIP uses the immediately previous device and account as default. All files with BAK extensions from account [2,2] on magtape unit Ø are transferred to the current user account on RK unit 1. The files are created with a protection code of <6Ø>. (The magtape on unit Ø is rewound before the search for the BAK files begins.)

The following command shows the default for a null input specification.

```
#MTØ:[5,5]<,DKØ:/BL:1Ø24
```

PIP transfers all files from the current user account in the public structure and all files from the current user account on RK unit Ø. Each input file is created on magtape unit Ø with a project-programmer number of 5,5 and with the default protection code. The output records are written in 1Ø24-byte blocks.

The following command shows the use of the /DOS option.

```
#DK1:<DTØ:,DKØ:[128,128]*.MAC/DOS,SY:*.BAS/FA
```

All files from DECtape unit Ø; all files with MAC extensions under UIC 2ØØ,2ØØ (octal) on the DOS disk from RK unit Ø; and all files with BAS extensions under account 128,128 in the public structure are transferred to the RSTS/E disk on RK unit 1 with the default protection code. The /DOS switch applies only to RK unit Ø; the UIC must be specified in decimal. Since no account specification appeared in the last file specification, the previous account specification is used. Files from DECtape unit Ø and from the public structure are transferred in formatted ASCII mode. Files from RK unit Ø are transferred in contiguous mode because they come from a DOS disk.

The /IN option with a file specification causes PIP to list, one at a time as a query, the files eligible for transfer. This feature allows the user to inspect each eligible file and to selectively transfer each file according to the response given to the query.

The /IN option must appear with each specification for which the inspect feature is to apply. For example,

```
#*.*<DKØ:[5,5]*.BAS/IN,*.SRT,*.CBL/IN/FA
```

In the command above, files with BAS, SRT, and CBL extensions on RK unit Ø under account [5,5] are to be transferred to the public structure under the current user account. The inspect feature applies to files with BAS and CBL extensions because of the accompanying /IN option. To transfer an eligible file, type YES (or any string beginning with Y) in response to the inspect query. To omit an eligible file from the transfer, type NO (or any string not beginning with Y). Files with SRT extensions are transferred without the inspect feature being applied.

4.18.4 Wild Card Specifications in Rename Commands

For rename commands, extended PIP does not apply the standard defaults in all cases. In the output, device and account defaults are taken from those applied in the input and the protection code is that of the input file. The device and account, therefore, need not be repeated in the output specification. For example,

```
#*.TMP=DKØ:[5,5]*.BAS/RE
```

PIP renames all files with BAS extensions in account 5,5 on RK unit Ø. Each output file has the same name and protection code as the input file but is given the extension TMP. No transfer occurs and only files on RK unit Ø are affected.

Except for protection code, the defaults in the input of a rename command are the same as those described in Table 4-28. The protection code of input files is preserved unless otherwise explicitly specified. The following example shows the device and account defaults applied on input.

```
#*.TMP=*.SRT,DKØ:[5,5]*.SRT,DK1:*.SRT/RE
```

Extended PIP renames all files with SRT extensions under the current account in the public structure; all files under account [5,5] on RK unit Ø: and all files with SRT extensions under account [5,5] on RK unit 1. The device and account defaults are applied whenever an explicit element is not given.

If the null specification is given as input, all files in the current account on the public structure are used. For example,

```
#.TMP=/RE
```

PIP renames all files and uses the name of the input file but gives the extension TMP. If a file of the same name exists currently, PIP prints an error message but continues the rename operation.

The following command shows the defaults used when the protection code of files is changed.

```
#DK1:*.BAS<4Ø>/RE
```

The protection code of all files with BAS extensions under the current user account on RK unit 1 is changed to 4Ø. The original name and extensions are preserved because no name and extension is given in the output.

The /IN option in a rename command applies only to the file specification with which it appears. For example,

```
#*.BAK=DK1:*.BAS/IN,SY:*.BAS/RE
```

Only the eligible files under the current account on RK unit 1 are listed for inspection. The eligible file is renamed only if YES (or any string beginning with Y) is typed in response to the inspect query. If NO (or any string not beginning with Y) is typed, the file is not renamed. After all eligible files on RK unit 1 are processed, the eligible files under the

current account on the public structure are renamed without the inspect action. If the inspect action is desired for the second specification also, the /IN option must appear twice as shown below.

```
#DK1:*.BAS/IN,SY:*.BAS/IN/RE
```

As a result, the inspect action is applied to both specifications.

4.18.5 Wild Card Specifications in Delete Commands

For delete commands in extended PIP; device and account defaults are taken from the immediately previous device and account specified. If a device or an account is not explicitly specified, PIP applies the device and account defaults described in Table 4-28. The file name and extension must be explicit; extended PIP applies no defaults for either. The following commands show the proper procedure.

```
#DK1:[5,5]A.DAT,B.BAS/DE
```

The files A.DAT and B.BAS in account 5,5 on RK unit 1 are deleted. The device and account defaults are applied for the specification B.BAS. The following command shows the defaults applied in absence of an explicit device and account.

```
#A.DAT,B.BAS/DE
```

The files A.DAT and B.BAS are deleted from the current user account in the public structure. The following command shows the usage of the immediately previous device specification.

```
#ABC.*,DT1:ABC.*,*.ABC/DE
```

All files with name ABC are deleted from the public structure and from DECtape unit 1. For the third specification, PIP applies the immediately previous device default and deletes all files with extension ABC on DECtape unit 1.

If the file name and extension are not explicit in a delete command, PIP prints an error message and terminates the operation. For example, the following commands generate errors.

```
/DE        (file name and extension is missing)
.*/DE      (file name is missing)
*/DE       (extension is missing)
```

The /IN option in delete commands applies only to the file specification with which it appears. For example,

```
#DK1:*.BAS/IN,SY:*.BAS/DE
```

Only the eligible files under the current account on RK unit 1 are listed for inspection. The eligible file is deleted only if YES (or any string beginning with Y) is typed in response to the inspect query. If NO (or any string not beginning with Y) is typed, the file is not deleted. After all eligible files on RK unit 1 are processed, the eligible files under the current account on the public structure are deleted without the inspect action. If the inspect action is desired for the second specification also, the /IN option must appear twice as shown below.

```
#DK1:*.BAS/IN,SY:*.BAS/IN/DE
```

As a result, the inspect action is applied to both specifications.

4.18.6 Processing ANSI Magtape Files

Extended PIP handles ANSI D, ANSI U, and ANSI F formats on input and ANSI D and ANSI U formats on output. Table 4-30 shows the transfers possible.

Table 4-30
Possible ANSI Magtape Transfers

Output	Input
ANSI D ANSI U	ANSI D
ANSI D ANSI U	ANSI U
ANSI D ANSI U ANSI F	ANSI F

Unless otherwise specified, Extended PIP transfers a magtape file in the format in which it was written. For example,

```
#MT1:ABC.DAT<MTØ:ABC.DAT
```

The input file from unit 0 is created on the output unit 1 in the same format in which it was written.

The format of the output file is altered if an option appears with the file specification. Again, the only changes in format allowed are those shown in Table 4-30. To force Extended PIP to read the input file as ANSI U, the /BLOCK option is used. The /BLOCK option forces the input file to be read as if it were written in ANSI U format. The option must appear with the file specification to which it applies. For example,

```
#SY:OUT<DKØ:INP1,MTØ:INP2/BLOCK,MT1:INP3/BL:1Ø24
```

The output file SY:OUT is written with a block size of 1024. The disk file INP1 is read with a block size of 1024. Because of the /BLOCK option, the magtape file INP2 is read as if it were written in ANSI U format with a maximum block size of 1024; that is, INP2 is read using GET statements without deblocking. The magtape file INP3, however, is read using the block size and record size with which it was written. If INP3 is in ANSI D format (variable record length), each record is deblocked to occupy one 1024-byte block on the disk. If INP3 is in ANSI U format (fixed record length), each block of input occupies one 1024-byte block on the disk. The /BLOCK option, therefore, causes files to be transferred more quickly since deblocking is not performed.

The /BLOCK applies only to the file with which it appears. For example,

```
#SY:OUT<DKØ:INP1,MTØ:INP2/BLOCK/FA
```

The output file SY:OUT is written in formatted ASCII mode. The input file INP1 is read as a formatted ASCII file and INP2 is read as an ANSI U format file.

The /FORMAT:U and /FORMAT:V force the magtape output characteristics to the undefined format and the variable length record format, respectively. If neither switch appears, the files are processed according to the default assigned to the drive. For example,

```
#MTØ:<MT1:
```

If the input unit is not assigned as ANSI, the output files are written in ANSI U format. If the input drive is assigned as ANSI, each output file is written in the same format as the corresponding input file. Specifying an output file name causes the input files to be merged into one output file having the characteristics of the first selected input file. Table 4-31 summarizes the effect of drive assignments in the above command.

Table 4-31
Effect of Labeling Default Assignments

Input	Output	Effect
ANSI	ANSI	Output files are copied exactly from the input volume. All volume and file header information is copied.
ANSI	DOS	Output files are created as if in a contiguous mode transfer. No file attributes are copied because DOS does not support file attributes. Input files are read in the format in which they were written. (For example, an ANSI D file is read as a variable length record file.)
DOS	ANSI	Output files are written in ANSI U format.
DOS	DOS	A formatted ASCII (the default mode in PIP) transfer is performed.

(For information on assigning a labeling format to a magtape drive, see Section 2.7.4.)

To force an output file to have either undefined or variable length record format, specify either the /FORMAT:U or the /FORMAT:V switch in the output.

4.18.7 Indirect Command Files in Extended PIP

Extended PIP can execute commands from a file. Such a file is termed an indirect command file and is indicated by the commercial at (@) character. The commands in the file are executed when an indirect command is given. An indirect command has the @ character as the first character on the command line followed by the file specification. The following is the correct format for an indirect command.

```
@[proj,prog]dev:file.ext
```

The defaults are the current account in the public structure and the extension CMD. A file name must be specified in the command. The device can be either disk or DECtape.

The indirect command file is an ASCII file containing extended PIP commands. If a line of the file begins with a semi-colon (;) character, PIP treats it as a comment line and skips to the next line of the file. The REALLY ZERO and inspect questions of the /ZE and /IN options are not printed when executed from an indirect command file.

The indirect command can appear in the indirect command file. Ten levels of nesting are allowed. If more than 1Ø levels are attempted, the program prints the message INDIRECT COMMAND ERROR - COMMAND STACK OVERFLOW.

Most errors cause PIP to return to the base level - the level at which the first indirect command file was initiated. On the following errors, however, PIP simply prints the error message and continues processing.

NO FILES MATCHING specification for /DE operations
NO FILES MATCHING specification for /LI operations

where specification is the full file specification

FILE OR ACCOUNT ALREADY EXISTS for /RE operations

The following command shows the procedure to execute an indirect command file.

#@ABC

PIP reads the file named ABC.CMD in the current user account in the public structure. If only the designator KBn: appears in the indirect command, PIP reads the specified keyboard for commands. Execution of indirect commands from the keyboard is terminated when the CTRL/Z combination is typed.

Since the logically assigned account (see Section 2.7.1) and the PIP indirect command are denoted in the same manner, the placement of the @ character on the PIP command line is important. For example, to obtain a directory listing of the logically assigned account, the @ character must be used with the /LI option. Because the @ character must be followed by a file name, the command @/LI generates an error. The following command, however, executes properly.

#SY:@/LI

The directory of the logically assigned account in the public structure is generated.

4.18.8 Extending the Physical Command Line - /MORE

The option /MORE at the end of a command line causes PIP to print the text MORE > and space to the next tab position. The command line can thereby be continued on the next physical line. The entire command is not executed until /MORE does not appear in the command line. For example,

```
#LPØ:=A.DAT,B.BAS,C.BAC/MORE
MORE > D.BAK/LI
```

PIP executes the /LI option for the files A.DAT, B.BAS, C.BAC, and D.BAK.

CHAPTER 5

RSTS-11 PERIPHERAL DEVICES

RSTS-11 has several peripherals which are available to the user. These devices include:

user terminal (ASR-33, ASR-35, LA3Ø & LA36 DECwriters, VTØ5 & VT5Ø displays)
high-speed paper tape reader/punch
card reader
line printer
DECtape
magtape
floppy disk

While normal operation of a computer system is by programmed control, manual operation is necessary for some tasks. This chapter describes the manual control and operation of the common RSTS-11 user peripherals.

5.1 ASR-33 TELETYPE

The ASR-33 Teletype is an inexpensive, commonly employed user terminal. Major features are noted in Figure 5-1.

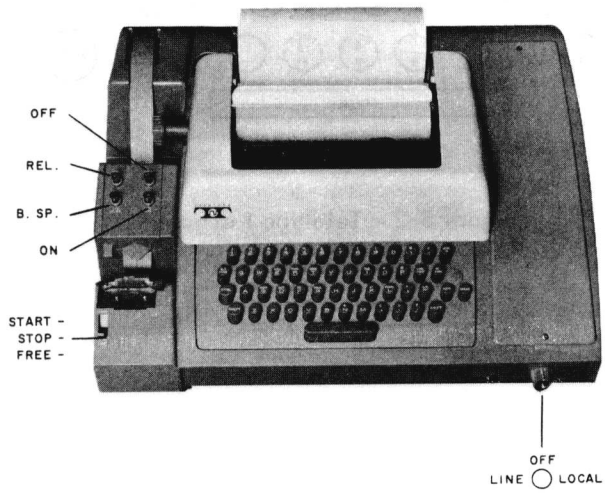


Figure 5-1 ASR-33 Teletype Console

The components of the Teletype unit and their functions are described on the following page.

5.1.1 Control Knob

The control knob of the ASR-33 Teletype console has three positions:

LINE	The console has power and is connected to the system as an I/O device under RSTS-11 control.
OFF	The console does not have power.
LOCAL	The console has power for off-line operation under control of the keyboard and switches only.

5.1.2 Keyboard

The Teletype keyboard shown in Figure 5-2 is similar to a typewriter keyboard, except that some nonprinting characters are included as upper case elements. For typing characters or symbols such as $, %, or # which appear on the upper portion of numeric keys and some alphabetic keys, the SHIFT key is depressed while the desired key is typed.

Nonprinting operational functions are shown on the upper part of some alphabetic keys. By depressing the CTRL (control) key and typing the desired key, these functions are activated. Use of these CTRL/key combinations are described in Chapter 3 of this manual.[1]

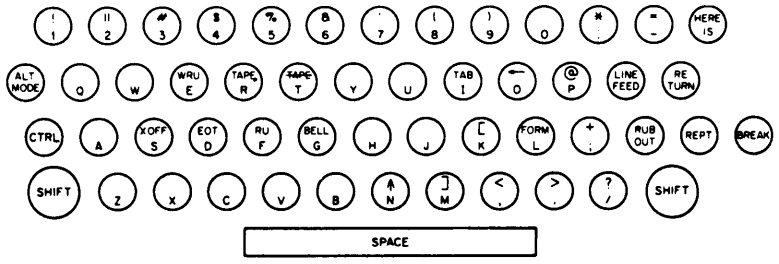


Figure 5-2 Teletype Keyboard

[1]Although not shown on most keyboards, SHIFT/L produces the backslash character (\) and SHIFT/K and SHIFT/M produce the square brackets, [and], respectively.

5.1.3 Printer

The Teletype printer provides a printed copy of input and output at ten characters per second, maximum rate. When the Teletype unit is on-line to the system (control knob turned to LINE), the copy is generated by the computer (even the echoing of the characters typed by the user); when the Teletype unit is off-line (control knob turned to LOCAL), the copy is generated directly from the keyboard onto the printer as a key is struck.

5.1.4 Low-Speed Paper Tape Reader

The paper tape reader is used to read data punched on eight-channel perforated paper tape at a rate of 1Ø characters per second, maximum. The reader controls are shown in Figure 5-1 and described below.

START	The reader is activated; reader sprocket wheel is engaged and operative.
STOP	The reader is deactivated; reader sprocket wheel is engaged but not operative.
FREE	The reader is deactivated; reader sprocket wheel is disengaged.

The following procedure describes how to properly position paper tape in the low-speed reader.

a. Raise the tape retainer cover.

b. Set reader control to FREE.

c. Position the leader portion of the tape over the read pins with the sprocket (feed) holes over the sprocket (feed) wheel and with the arrow on the tape (printed or cut) pointing outward (forward).

d. Close the tape retainer cover.

e. Make sure that the tape moves freely (if the tape does not move back and forth freely, the paper feed holes are not properly positioned).

f. Set reader control to START, and the tape is ready.

5.1.5 Low-Speed Paper Tape Punch

The paper tape punch is used to perforate eight-channel rolled oiled paper tape at a maximum rate of 1Ø characters per second. The punch controls are shown in Figure 5-1 and described on the following page.

RELease	Disengages the tape to allow tape removal or loading.
B.SP	Backspaces the tape one space for each firm depression of the B.SP button.
ON (LOCK ON)	Activates the punch.
OFF (UNLOCK)	Deactivates the punch.

Blank leader/trailer tape is generated by:

a. Turning the control knob to LOCAL.
b. Turning the punch control to ON.
c. Typing the HERE IS key.
d. Turning the punch control to OFF.
e. Turning the control knob to LINE.

5.2 HIGH-SPEED PAPER TAPE READER AND PUNCH UNITS

One high-speed paper tape unit can be provided with each RSTS-11 system. This unit is mounted on the central computer console. A high-speed paper tape unit is pictured in Figure 5-3 and descriptions of the reader and punch units follow.

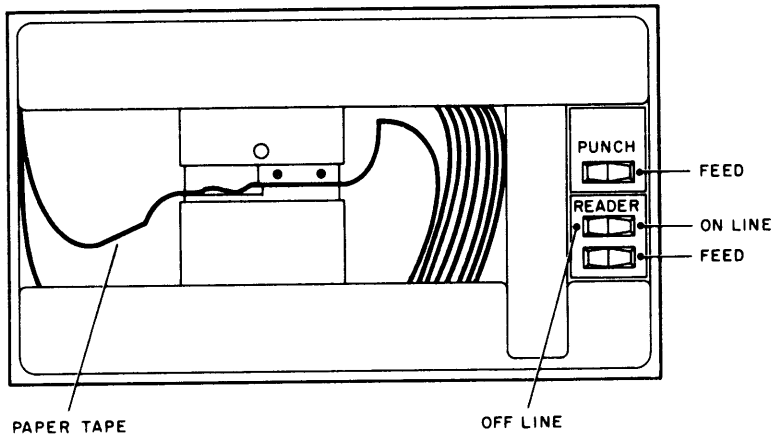


Figure 5-3 High-Speed Paper Tape Reader/Punch

5.2.1 High-Speed Reader Unit

The high-speed paper tape reader is used to read data from eight-channel, fan-folded (non-oiled), perforated paper tape photoelectrically at a rate of 3ØØ characters per second, maximum.

NOTE

Tape from the Teletype punch should not be used with the high-speed reader as the oil on the tape causes lint and dust to collect on the photoelectric cells.

Primary power is applied to the reader when the computer power is on.

In order to use the high-speed reader as an input device, turn the reader ON LINE/OFF LINE rocker switch to ON LINE. Load tape into the reader as explained below:

a. Raise the tape retainer cover.

b. Place tape in right-hand bin with printed arrows pointing toward left-hand bin. (Channel one of the tape is toward the rear of the bin.)

c. Place several folds of blank tape past the reader and into the left-hand bin.

d. Place the tape over the reader head with feed holes engaged in the teeth of the sprocket wheel.

e. Close the tape retainer cover.

f. Depress the FEED rocker switch until leader tape is over the reader head.

The reader is capable of sensing whether a tape is in the reader. If an attempt is made to read a tape when the reader is either empty or OFF LINE, an error is generated.

5.2.2 High-Speed Punch Unit

The high-speed paper tape punch is used to record computer output on eight-channel, fan-folded, non-oiled paper tape at a rate of 5Ø characters per second maximum. All characters are punched under program control from the computer. Blank tape (feed holes only, no data), is produced by pressing the punch FEED rocker switch. The punch unit has power turned on whenever the computer has power; it does not require any additional on/off switch.

Fan-folded paper tape is generally grey in color. When a box of tape is nearly empty, purple tape is produced. Rather than risk running out of tape while punching, replace the box of paper tape at this point or notify the system manager who will replace the tape.

5.3 CR11 CARD READER

The CR11 card reader allows the RSTS-11 system to accept information from punched 8Ø column data cards. Cards can be read under program control at rates of up to 2ØØ or 3ØØ cards per minute.

Power to the reader, shown in Figure 5-4, is controlled by the ON/OFF switch on the upper left hand corner of the back panel. Two toggle switches are present on the back panel and should be set to AUTO and REMOTE for proper operation. The LAMP TEST button on the reader back panel can be depressed to check the operation of the various reader lights on the front panel.

In order to use the card reader:

a. Remove card weight from input hopper. Place cards loosely in input hopper. The first card to be read is placed at the front of the deck, "9" edge down, column 1 to the left. Replace card weight on top of cards in input hopper. Cards should not be packed tightly.

b. Press green RESET button. Wait 4 seconds for RESET light to come on. The card deck is now able to be read under program control.

c. Cards may be loaded while the reader is operating provided tension is maintained on the front of the deck as cards are added to the rear. Additional cards should not be loaded until the hopper is 1/2 to 2/3 empty.

d. The output stacker bin can be unloaded while cards are being read. Care should be taken to maintain the order of the deck.

The various lights and switches (buttons) on the reader front panel and their significance are described in Table 5-1.

Table 5-1

Card Reader Controls

Light/Switch	Function
POWER	light indicates that there is power to the reader.
READ CHECK	light indicates a reading error, torn card, or card too long for reader. Reader stops and RESET light is out.
PICK CHECK	light indicates inability to remove card from input hopper. Reader stops and RESET light is out.
STACK CHECK	light indicates inability to remove card from input hopper. Reader stops and RESET light is out.

Table 5-1 (cont'd.)

Light/Switch	Function
HOPPER CHECK	light indicates that there are no cards in input hopper or the output stacker is full. Condition must be manually corrected to allow further operation.
STOP	button, when depressed, causes red light to go on momentarily. RESET light goes out and reader operation stops as soon as the card currently in the read station has been read.
RESET	button, when depressed, causes green RESET light to go on and initializes card reader logic.

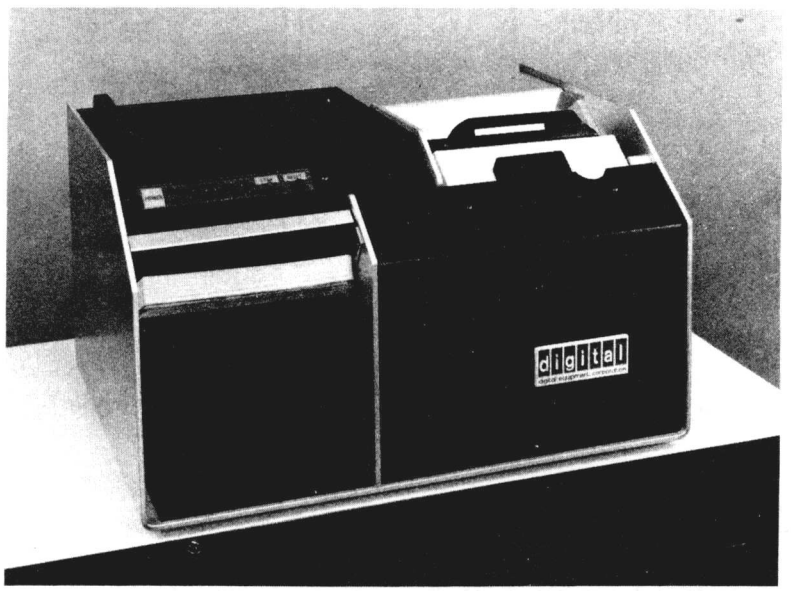


Figure 5-4 CR11 Punched Card Reader

5.4 LP11 LINE PRINTER

The LP11 line printer, pictured in Figure 5-5, has an 8Ø column capacity, prints at a rate of 356 lines per minute at a full 8Ø columns, and can print 11ØØ lines per minute at 2Ø columns. These rates are based on a 64-character set. A 96-character set and 132-column version are also available. The print rate is dependent upon the data and number of columns to be printed.

Characters are loaded into a 2Ø-character printer memory via a Line Printer Buffer. When this buffer is full, the characters are automatically printed. This process continues until the 8Ø columns (four print zones) have been printed or a carriage return, line feed, or form feed character is recognized. The printer responds only to codes representing the character set and three control characters. All other codes are ignored.

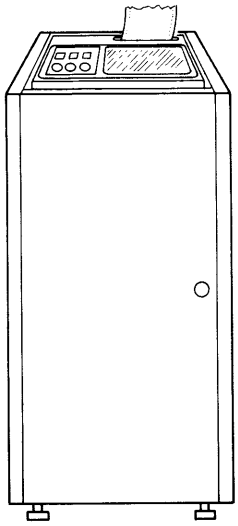


Figure 5-5 LP11 Line Printer System (8Ø-column model)

5.4.1 Line Printer Character Set

The 64 character set consists of the 26 upper case letters (A - Z), ten numerals (Ø - 9), 27 special characters and the space character. The 96-character set contains all of the above plus 26 lower case letters and 6 additional symbols. The character codes are 7-bit ANSCII.

Characters are printed 1Ø characters per inch and 6 lines per inch.

Line printers can use paper varying in width from 4 inches to 9-7/8 inches for the 8Ø-column printer. Forms making up to six copies can be used when multiple copy printing is desired.

The special symbols available are as follows:

64-character set	96-character set
! " # $ % & ' ( ) * + , - . / ; : < > = ? @ \ [ ] ^ _	all of the 64-character set symbols ` { } ¦ ~ DEL

ASCII numeric equivalents for the various characters are contained in Appendix D.

5.4.2 Line Printer Operation

Figure 5-6 illustrates the line printer control panel on which are mounted three indicator lights and three toggle switches. Operation of these switches and the power switch, and the meaning of the lights is explained in Table 5-2.

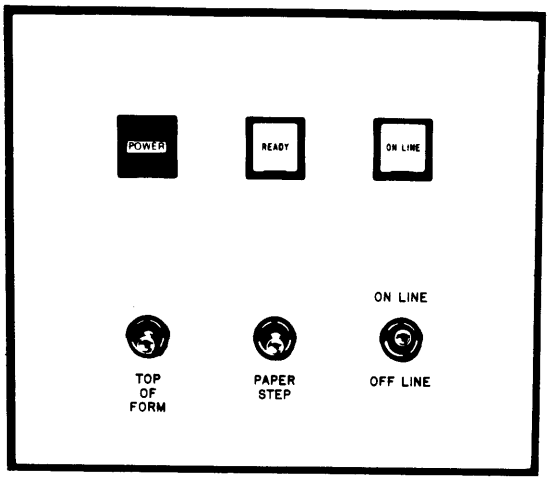


Figure 5-6 Line Printer Control Panel

Table 5-2

Line Printer Controls

Light/Switch	Function
POWER light	Glows red to indicate main power switch (located inside cabinet) is at ON position and power is available to the printer.
READY light	Glows white, shortly after the POWER light goes on to indicate that internal components have reached synchronous state , paper is loaded, and the printer is ready to operate.
ON LINE light	Glows white to indicate that ON LINE/OFF LINE toggle switch is in ON LINE position.

Table 5-2 (cont'd.)
Line Printer Controls

Light/Switch	Function
ON LINE/OFF LINE switch	This three-position toggle switch is spring-returned to center. When momentarily positioned at ON LINE it logically connects the printer to the computer and causes the ON LINE light to glow. Positioned momentarily at OFF LINE, the logical connection to the computer is broken, the ON LINE light goes off, and the TOP OF FORM and PAPER STEP switches are enabled. If printer is switched to OFF LINE, the ON LINE light remains on until either PAPER STEP or TOP OF FORM switch is activated. The printer should again be turned ON LINE.
TOP OF FORM switch	This two-position toggle switch is tipped toward the rear of the cabinet to roll the form to the top of the succeeding page. It is spring-returned to center position, and produces a single top-of-form operation each time it is actuated. The switch is effective when the printer is off line.
PAPER STEP switch	This two-position toggle switch operates similarly to TOP OF FORM but produces a single line step each time it is actuated. It is only effective when the printer is off line.
ON/OFF (main power) switch	This switch controls line current to the printer. To gain access to it, the printer front panel is unlatched, by pushing the circular button on the right hand edge, and opened to the left on its hinges. The switch is located to the left of center approximately fourteen inches below the top. If power is available, the red POWER light on the control panel will glow when the switch is positioned at ON. The switch is on when in the up position. The ON and OFF labels are printed on the stem of the switch. A group of two switches and three indicator lights, above the main power switch, are for the use of technicians in making initial adjustments to the printer.

The following procedure is used when loading paper in the line printer.

a. Open front door of cabinet. POWER light should be on. Printer should be off line.

b. Lift control panel TOP OF FORM switch and release to move tractors to correct loading position.

c. Open drum gate by moving drum gate latch knob to left and up. Swing drum gate open.

d. Adjust right hand tractor paper width adjustment for proper paper width if necessary. (Loosen set screw on 80 column printer; user release mechanism on 132 column printer.) Tighten tractor after adjustment.

e. Open spring loaded pressure plates on both tractors.

f. Load paper so that the two red arrows point to a perforation. Paper should lie smoothly between tractors without wrinkling or tearing the feed holes.

g. Close spring-loaded pressure plates on both tractors.

h. Adjust COPIES CONTROL lever to proper number of copies to be made, if necessary. Set to 1 or 2 for single forms, set to 5 or 6 for six-part forms.

i. Close drum gate and lock in position with drum gate latch. After 10 seconds the READY indicator should light.

j. Lift TOP OF FORM switch several times to ensure paper is feeding properly.

k. Set printer to on line. ON LINE indicator should light. At this point printed matter can be aligned with the paper lines, if desired, by rotating the paper vertical adjustment knob.

5.5 TC11/TU56 DECTAPE CONTROL AND TRANSPORT

DECtape units are available on most RSTS-11 systems. DECtape serves as an auxiliary magnetic tape storage facility to the system disk(s).

A DECtape peripheral unit consists of three components:

a. TU56 DECtape transport, pictured in Figure 5-7, which reads and/or writes information on magnetic tape.

b. TC11 Controller, the interface which controls information transfer. One controller serves up to four transports (up to 8 tape drives). The Controller is transparent to the RSTS-11 user.

c. DECtape, the recording medium used for data storage, consists of reel mounted magnetic tape formatted to permit read/write operations in either direction, error checking, block identification, and timing control.

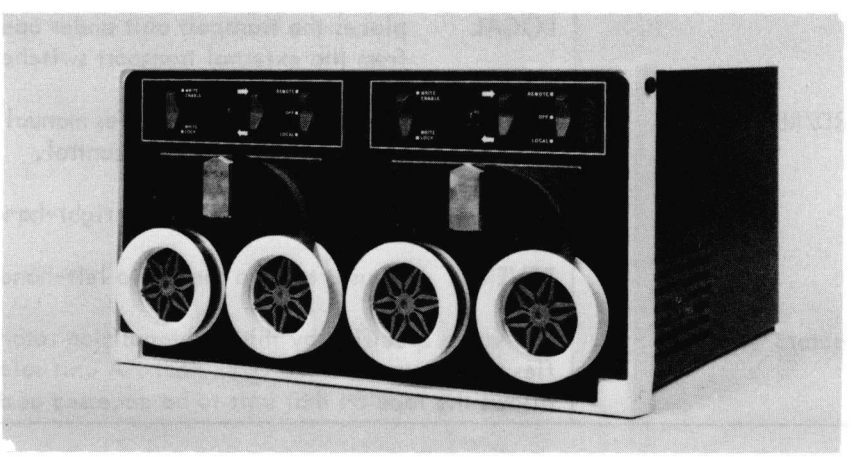


Figure 5-7 TU56 DECtape Transport

DECtape stores and retrieves information at fixed positions on magnetic tape. The advantage of DECtape over conventional magnetic tape is that information at fixed positions can be addressed. Conventional magnetic tape stores information in sequential (not addressable directly), variable-length positions. DECtape incorporates timing and mark information to reference the fixed positions. The ten-channel DECtape records five channels of information: a timing channel, a mark channel, and three information channels. These five channels are duplicated on the five channels remaining to minimize any possibility of information loss from the other channels.

Each formatted (certified) DECtape contains 578 blocks of data consisting of 256 (16-bit) PDP-11 words per block. 562 blocks are available to the user on each DECtape (several blocks are used for file directories).

Tape movement can be controlled by programmed instructions from the computer (i.e., through PIP) or by manual operation of switches located on the front panel of the transport. Data is transferred only under program control. The transport controls and lights are described in Table 5-3.

Table 5-3
DECtape Controls

Light/Switch	Function
REMOTE/OFF/LOCAL	This three-position rocker switch determines control of the DECtape unit. REMOTE enables computer control of the transport unit. OFF disables the transport unit. LOCAL places the transport unit under operator control from the external transport switches.
FORWARD/REVERSE	This two-position rocker switch enables manual winding of tape when transport unit is under LOCAL control. FORWARD causes tape to feed onto right-hand spool. REVERSE causes tape to feed onto left-hand spool.
Unit selector	The value specified by this eight-position rotary switch identifies the transport to the computer. A unit selector value of 1 allows the tape on that unit to be accessed as device DT1:

Table 5-3 (cont'd.)
DECtape Controls

Light/Switch	Function
WRITE ENABLE/WRITE LOCK	This two-position rocker switch determines whether or not a tape can be written. Any tape can be searched and read; however when the WRITE LOCK switch is on, the tape is protected from accidental writing or program deletion.
REMOTE light	When lit, the tape unit is on-line. The tape unit is on-line when: a. REMOTE/OFF/LOCAL switch is in REMOTE position, and b. unit selector switch setting agrees with the DECtape unit currently being accessed.
WRITE ENABLE light	When lit, indicates that the WRITE ENABLE/WRITE LOCK switch is set to WRITE ENABLE, regardless of whether the REMOTE light is on or not.

Operating procedures associated with DECtape units are described below. In order to mount a tape on a DECtape drive:

a. Set the REMOTE/OFF/LOCAL switch to OFF.

b. Place full DECtape reel on left spindle with label facing out. Press reel tightly onto spindle.

c. Pull tape leader over the two tape guides and the magnetic head until it reaches the take-up reel on the right hand side of the tape unit.

d. Wind loose tape end four turns around the empty right-hand reel by rotating right reel clockwise.

e. Set REMOTE/OFF/LOCAL switch to LOCAL. Verify that power is available to the tape unit.

f. Depress FORWARD/REVERSE switch in the FORWARD direction to wind about 15 turns onto the right-hand reel. This ensures that the tape is securely mounted.

In order to operate the DECtape unit on-line:

a. Set the REMOTE/OFF/LOCAL switch to either LOCAL or OFF and be sure power is available to the system.

b. Load the appropriate DECtape following the instructions above.

c. If the DECtape write operation is to be inhibited (to protect tape from accidental damage), set WRITE ENABLE/WRITE LOCK switch to WRITE LOCK. If the write operation is desired on this tape, set switch to WRITE ENABLE.

d. Dial the correct unit number on the unit selector. (No two active DECtape units should have the same unit number.)

e. Set REMOTE/OFF/LOCAL switch to REMOTE. Use of this DECtape unit is now under program control.

f. When on-line operation of this unit is to cease, set REMOTE/OFF/LOCAL switch to OFF or LOCAL. A moving tape can be stopped by quickly switching the REMOTE/OFF/LOCAL switch from REMOTE to LOCAL. The switch can be set to OFF when tape motion has stopped.

To remove a DECtape from the tape unit:

a. Set REMOTE/OFF/LOCAL switch to LOCAL.

b. Depress and hold REVERSE switch until all tape is wound onto the left-hand tape reel.

c. Set REMOTE/OFF/LOCAL switch to OFF.

d. Remove full reel from left-hand spindle.

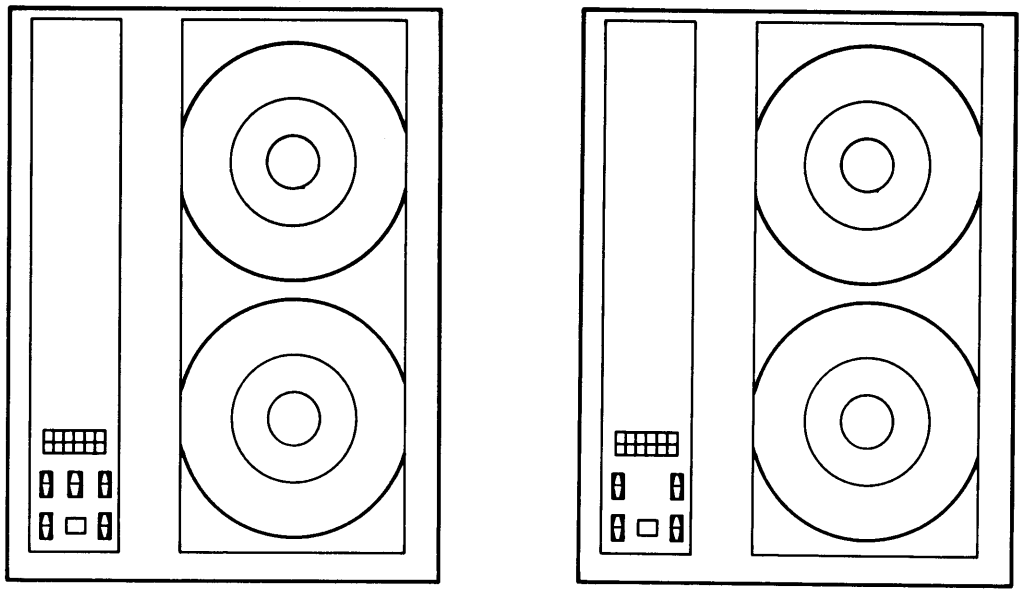
5.6 TM11/TU10 AND TJU16 MAGTAPE CONTROL AND TRANSPORT

Magtape is an optional addition to a RSTS-11 system. Magtape is used to provide storage for large volumes of data and programs. Writing, reading, and search operations are performed in a serial manner. Transfer of information can be made between RSTS-11 and other computer systems because TJU16 and TM11 Controllers read and write information in industry-compatible format.

The basic DECmagtape system consists of:

a. The TU10 Transport, pictured in Figure 5-8, can read or write information on magnetic tape in seven or nine channels (tracks). The TJU16 Tape Transport reads and writes information on nine channels only. The TU10 reads and writes data at 800 bits per inch (BPI), and the TJU16 Tape Transport can be used to read and write magtape either at 800 BPI or at 800 BPI and 1600 BPI.

b. The TM11 Controller is an interface between the TU10 Tape Transport and the PDP-11 system. The RH11 or RH70 is an interface between the PDP-11 and the TM02 Formatter/TU16 Transport. One controller, transparent to the RSTS-11 user, serves up to eight transport units.



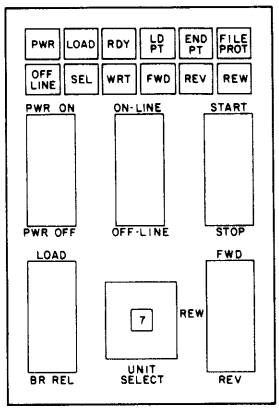
TU10 Transport TJU16 Transport

Figure 5-8 DEC Magnetic Tape System

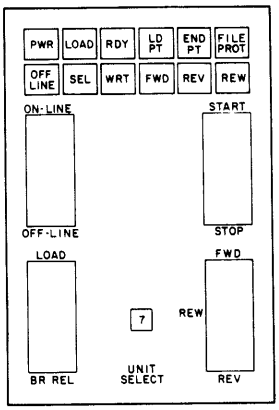
Transfer rate is up to 36,ØØØ characters per second for the TU1Ø and up to 72,ØØØ characters per second for the TJU16. Ten and one half inch magtape reels permit up to 24ØØ feet of tape per reel. Rewind time for a reel of 24ØØ feet is approximately 3 minutes, end to end.

5.6.1 Magtape Control Panel

The magtape transport control panel is shown in Figure 5-9. This panel is located at the lower left of the TU1Ø and TJU16 front panel shown in Figure 5-8. Table 5-4 describes the tape transport controls and Table 5-5 describes the various tape transport indicators.



TU10 Control Panel



TJU16 Control Panel

Figure 5-9 Control Panels

Table 5-4

Magtape Transport Controls

Switch	Function
PWR ON/PWR OFF	This two-position switch applies power to the TU10 Transport. (This switch does not exist on the TJU16 Transport.)
ON-LINE/OFF-LINE	This two-position switch controls operation of the transport unit. ON-LINE allows system operation under program control; OFF-LINE allows manual operation. Tape unit cannot be remotely selected when switch is in OFF-LINE position.
START/STOP	This two-position switch controls starting and stopping of tape motion. STOP does not stop transport during a rewind operation.
LOAD/BR REL	This three-position switch energizes the vacuum system in the LOAD position (necessary for any operation). The BR REL position releases vacuum tension and allows reels to be manually rotated. Center position locks reel brakes.
UNIT SELECT	This eight-position rotary switch identifies the transport to the computer. A unit select value of 1 allows the tape on that unit to be accessed as device MT1:. No two transports should be set to the same number.
FWD/REW/REV	This three-position switch moves the tape in the selected direction, depending on activation of the START/STOP switch. FWD moves tape forward until BOT or EOT marker is sensed. REW rewinds tape onto the feed reel until BOT marker is sensed. REV rewinds tape until BOT marker is sensed; toggling the START/STOP switch again causes the tape to rewind off the reel.

Table 5-5

Magtape Transport Indicators

Light	Function
PWR light (power)	When lit, indicates that power is available to the transport unit.
LOAD light	When lit, indicates that vacuum system has been enabled, allowing either on-line or off-line commands.
RDY light (ready)	When lit, indicates that all I/O lines are enabled. Transport can accept processor commands provided SEL light is also lit.
LD PT light (load point)	When lit, indicates that BOT marker has been sensed; transport ready for operation.
END PT light (end point)	When lit, indicates that EOT marker has been sensed; all tape motion stops to prevent tape from winding off reel.

Table 5-5 (cont'd.)

Magtape Transport Indicators

Light	Function
FILE PROT light (file protection)	When lit, indicates that writing on the tape is inhibited. This is true if no file reel is mounted on feed reel hub or if a file reel is mounted without a write enable ring.
OFF-LINE light	When lit, indicates that transport can be operated manually and cannot be operated under program control.
SEL light (select)	When lit, indicates that transport has been selected and is completely on-line. Transport can read or write data.
WRT light (write)	When lit, indicates that write-enable ring has been installed on feed reel and transport can write on tape.
FWD light (forward)	When lit, indicates tape is moving in forward direction.
REV light (reverse)	When lit, indicates that tape is moving in reverse direction.
REW light (rewind)	When lit, indicates that tape is being rewound; Tape continues until BOT marker is sensed.

5.6.2 Magtape Operating Procedures

Whenever handling magnetic tapes and reels, it is important to observe the following precautions to prevent loss of data and/or damage to tape handling equipment:

a. Handle a tape reel by the hub hole only. Squeezing reel flanges can damage tape edges when winding or unwinding tape.

b. Never touch tape between BOT and EOT markers. Do not allow end of tape to drag on floor.

c. Never use a contaminated reel of tape; this spreads dirt to clean tape reels and can affect transport operation.

d. Always store tape reels inside containers. Keep empty containers closed so dust and dirt cannot collect.

e. Inspect tapes, reels, and containers for dust and dirt. Replace old or damaged take-up reels.

f. Do not smoke near transport or tape storage area. Smoke and ash are especially damaging to tape.

g. Do not place transport near a line printer or other device that produces paper dust.

h. Clean tape path frequently.

To mount a tape reel on the magtape transport.

a. Apply power to the transport. (Depress the PWR ON switch on the TU10 transport.) Ensure that the LOAD/BR REL switch is in the center position and that the ON-LINE/OFF-LINE switch is in its OFF-LINE position.

b. Place a write-enable ring in the groove on the file reel if data is to be written on the tape. If writing is not required, be sure there is no ring in the groove.

CAUTION

Attempting to read a 7-track magtape reel mounted on a 9-track transport, or vice versa, causes the RSTS system and most other operating systems to crash. It is suggested that the user clearly label magtape reels and magtape transports as either 7-track or 9-track.

Two or more magtape transports set to the same unit number cause the RSTS system and most other operating systems to crash when an attempt is made to access a unit by that unit number. It is suggested that the unit numbers of all magtape transports be set in some consistent order and that they be clearly labelled and never changed.

c. Mount file reel onto lower hub with groove facing toward the back. Press reel tightly onto spindle; tighten center nut.

d. Install take-up reel (top reel), if necessary, as described in (c) above. The top reel is generally permanent and should not require installation by the user.

e. Place LOAD/BR REL switch to the BR REL position.

f. Unwind tape from the file reel and thread tape over tape guides and head assembly as shown in Figure 5-10. Wind about five turns of tape onto take-up reel.

g. Set LOAD/BR REL switch to LOAD position to draw tape into vacuum columns. As a result, the LOAD light comes on.

h. Select FWD and depress the START switch to advance the tape to the load point. When BOT marker is sensed, tape motion stops, the FWD indicator goes out and the LOAD PT indicator comes on.

   If tape motion continues for more than 10 seconds, depress STOP, select REV, and then depress START. The tape will advance to the BOT marker before stopping. (This may be necessary if, in winding the tape manually, the BOT marker has already been passed.)

Setting the ON-LINE/OFF-LINE switch to ON-LINE allows the transport to accept commands from the controller under program control. The transport is not fully on-line until the RDY and SEL indicators are lit.

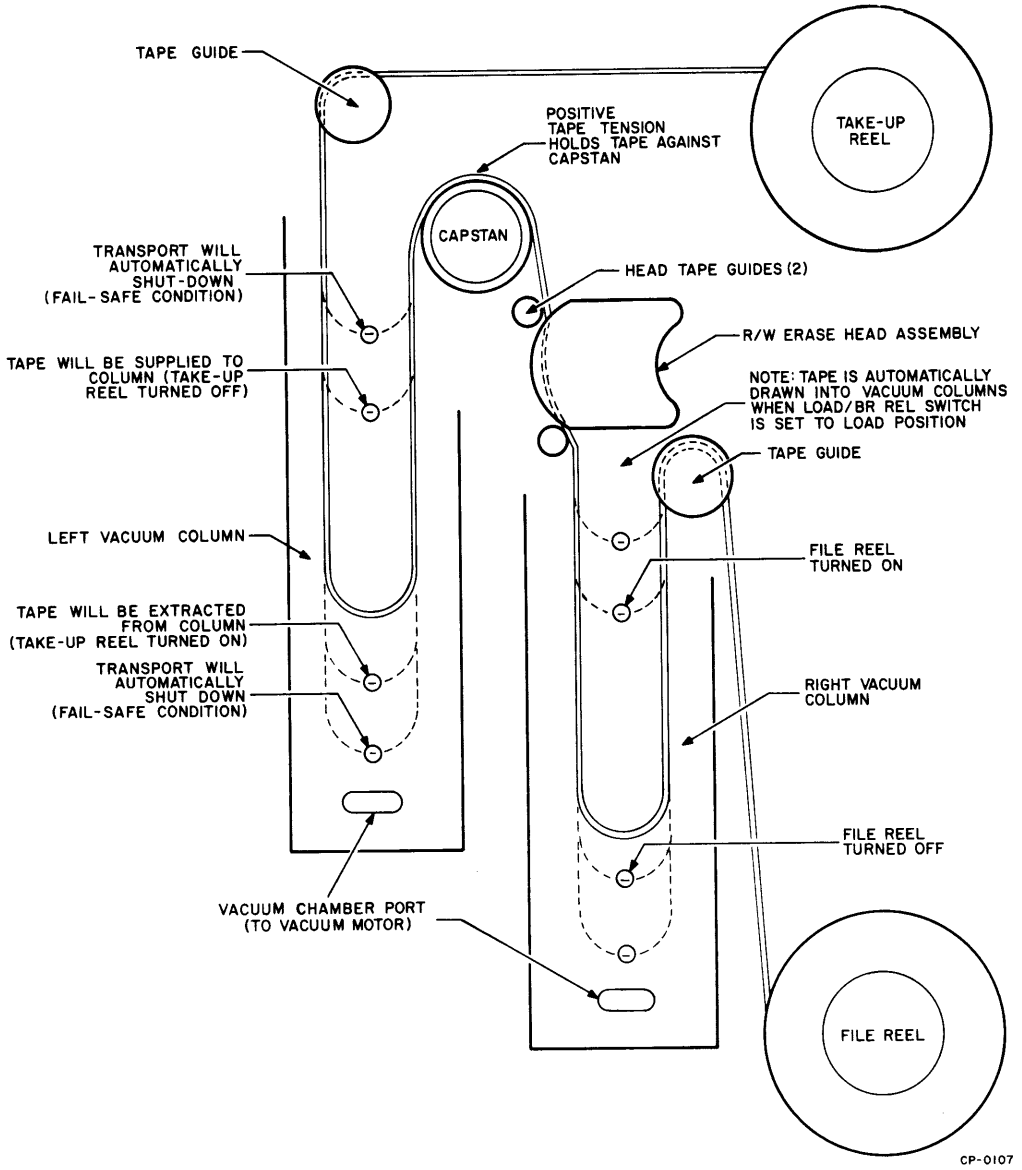


Figure 5-10 Magtape Transport Threading Diagram

To remove a tape from the transport unit:

a. Set ON-LINE/OFF-LINE switch to OFF-LINE position.

b. Set START/STOP switch to STOP position.

c. Set FWD/REW/REV switch to REW position.

d. Set START/STOP switch to START position Tape rewinds until BOT marker is reached.

e. Set LOAD/BR REL switch to BR REL position to release brakes.

f. Gently hand wind the file reel in a counter clockwise direction until all of the tape is wound onto the reel. Do not jerk the reel. This may stretch or compress the tape which can damage data.

g. Remove the file reel from the hub assembly.

5.7 VTØ5 ALPHANUMERIC DISPLAY TERMINAL

The VT05 alphanumeric display terminal consists of a cathode ray tube (CRT) display and a self-contained keyboard. The VT05 keyboard operates as a typewriter keyboard except that no hard copy is produced. Each graphic character generated by typing at the keyboard is converted to a 5 by 7 dot matrix and displayed on a television-like screen. The full capacity of the screen is 20 lines, each containing 72 character positions for a total of 1440 characters.

When power is applied to the terminal and the CRT filament is warmed up, a blinking indicator called the cursor appears at the leftmost position of the top line (line number 1) of the screen. The blinking cursor indicates the position which the next generated character will occupy on the screen. The cursor can be moved up, down, left, or right by the use of various control characters generated by keys located to the right of the keyboard. When a displayable character is generated, its representation is displayed and the cursor automatically moves right to the next character location until the cursor reaches character position 72.

A speaker in the VT05 emits an audible tone or beep when the cursor reaches character position 65. This action warns the user that the cursor is within 8 spaces of the end of the line. (The speaker also beeps when the terminal or the computer generates the BEL character.) When the cursor reaches position 72 on a line, characters subsequently generated replace the character previously in position 72. Automatic carriage return or line feed is a hardware modification and terminals without the modification must be programmed to include these automatic operations.

The VT05 keyboard, shown in Figure 5-11, is similar to a typewriter keyboard except for the following operations keys:

ALT	Generates the ESC character CHR$(27).
CTRL	Control key is used to generate various control character combinations. See Chapter 3 for a description of control characters.
LF	Generates the LINE FEED character CHR$(10).
CR	Generates the RETURN character CHR$(13).
RUBOUT	Generates the DEL character CHR$(127).
TAB	Generates the HT character CHR$(9) and causes the cursor to move to the right to the next tab stop. Tab stops are preset eight character spaces apart and are at locations 1, 9, 17, 25, 33, 41, 49, 57 and 65. Once the cursor reaches character position 65, the HT character moves the cursor right one position. Another HT character causes a RETURN and LINE FEED, leaving the cursor at position 1.

A line feed typed with the cursor in the bottom line (line 20) causes all displayed data on the screen to move up one line and any data in the top line to disappear. This action is termed automatic scrolling and is useful when a large amount of data is transmitted from the computer to the VT05 and is to be received and displayed. The user can employ the CTRL/S combination described in Chapter 3 to temporarily suspend transmission of such output to the screen and enable examination of data currently displayed on the screen. The CTRL/Q combination resumes output.

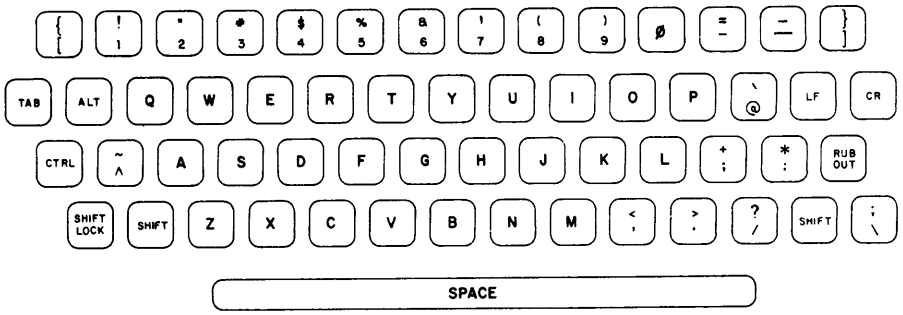


Figure 5-11 VT05 Keyboard

Four actions erase characters from the screen. A character is erased when the cursor is placed under it and a space character is generated. A character is replaced on the screen if the cursor is positioned under an existing character and another character is generated. The EOL and EOS special functions also erase characters from the screen.

The DEL code, CHR$(127), generated by typing the RUBOUT key, is ignored and no visual indication occurs on the display. When the RUBOUT key is typed on a VT05 terminal directly connected to a RSTS-11 system, the system backspaces the cursor one character position, generates a space character in that position on the screen and in memory and backspaces one character position again. A RUBOUT key typed on a VT05 terminal connected to a RSTS-11 system by a dial up line is treated as the RUBOUT key typed at a teleprinter unless the TTYSET SCOPE command is in effect for that line. Executing the SCOPE command causes the RUBOUT key to be treated as described above.

The ALT key typed at a VT05 generates the ESC character. When received by the VT05, the ESC character has no effect on the display. RSTS-11 system programs such as GRIPE treat the ESC character as a line terminating character; the system echoes the $ character on the display when the ESC character is received.

Controls to adjust the quality of the display are located on the right hand side of the terminal shown in Figure 5-12. Means to select a baud rate and mode of operation are on the rear panel of the device. At speeds above 300 baud, FILL characters for time delay as shown in Table 5-6 are required after some control characters are generated. The number of FILL characters required depends upon the baud rate at which the terminal operates.

Table 5-6

FILL Characters Required for VT05

Baud Rate	Number of FILL Characters
300	none
600	1
1200	2
2400	4

To effect the proper number of FILL characters on a VT05 terminal which operates above 300 baud and whose characteristics are not permanently set, use the FILL command of the TTYSET system program.

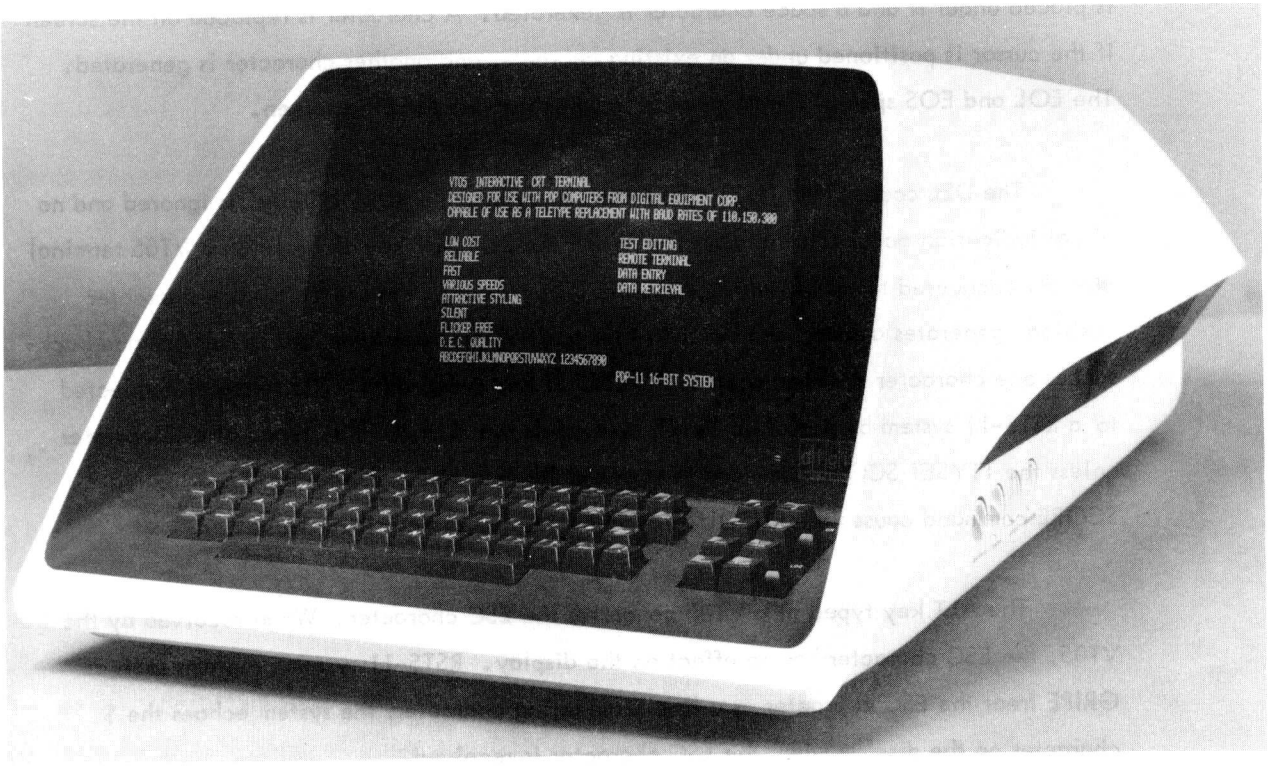


Figure 5-12 VT05 Alphanumeric Display Terminal

5.7.1 Controls and Operating Procedures

The controls and switches for a VT05 terminal are listed and described in Table 5-7. To start the terminal connected by a direct line to the RSTS-11 system, do the following.

a. Set the LOC/REM switch to the REM position.

b. Set the ON/OFF switch to the ON position. Allow approximately one minute for the filament to warm up, for the blinking cursor to appear in the home position (upper left corner of the display), and for the speaker to emit one beep.

c. If the cursor fails to appear as specified, press the HOME key. Ensure that the BRIGHTNESS control is not turned fully counterclockwise. If the cursor still fails to appear, place the ON/OFF switch in its OFF position and report the malfunction to the proper person.

d. To properly adjust the clarity of the display, turn the CONTRAST control counterclockwise to its minimum setting. Turn the BRIGHTNESS control counterclockwise until the characters are barely visible. Adjust the CONTRAST control to the optimum level.

e. When the operating session is completed, set the ON/OFF switch to its OFF position.

To operate the VT05 terminal which is connected to the computer by a dial up line, perform the following steps.

a. Follow the procedures to start the VT05 terminal as if it were connected by a direct line to the RSTS-11 system and set the BAUD RATE selector switch on the rear panel to its 110 position or to the position required by the permanent default characteristics established by the system manager for that line.

b. Dial the RSTS-11 system and perform the log in procedures. If the line does not have permanent default characteristics for a VT05, continue with step c. Otherwise, go to step e.

Table 5-7

VT05 Controls and Switches

Location	Label	Operation
Keyboard	ON/OFF	When placed in its ON position, power is applied to the VT05 display and the refresh memory is cleared. When placed in its OFF position, the VT05 is inoperative and the screen is darkened.
	LOC/REM	When placed in its LOC position, it breaks the electrical connection between the device and the computer. Local operation for maintenance and training is still allowed. In its REM position, it connects the terminal to the remote computer system. Data is simultaneously transmitted to and received from the computer.
Right Side	BRIGHTNESS	Turning this control in the clockwise direction increases the brightness of the screen. Turning the control counterclockwise decreases brightness. If turned fully counterclockwise, the display is completely darkened.
	CONTRAST	This control increases and decreases the clarity of the characters displayed on the screen.
	VERTICAL	This control synchronizes the display in the vertical position such that all 20 lines are visible on the screen.

Table 5-7 (Cont).
VT05 Controls and Switches

Location	Label	Operation
	HORIZONTAL	This control moves the display in the horizontal direction such that all 72 character positions are visible on the screen.
Rear Panel	BAUD RATE	This 10-position selection switch determines the rates at which the terminal transmits and receives data.
	FULL/HALF DUPLEX	When at FULL, it allows the keyboard to transmit data to the computer and allows the display to simultaneously receive data from the computer. When at HALF, data is transmitted to the VT05 receiver logic as well as to the computer.

c. Run the TTYSET system program and type the VT05 macro command or the SPEED command with the proper baud rate value. (The maximum baud rate allowed on a voice grade telephone line is 300.)

d. Set the BAUD RATE selector switch on the rear panel to the proper value, after which the VT05 operates at the proper baud rate.

e. When the operating session is completed, set the BAUD RATE selector switch to the 110 position and set the ON/OFF switch to its OFF position.

5.8 2741 COMMUNICATIONS TERMINALS[1]

RSTS-11 systems allow the use of 2741 compatible[2] terminals for time sharing operations. 2741 terminals employ a Selectric typing mechanism which provides high quality copy in upper and lower case format. Such copy is especially suitable for documentation preparation and for output of the RUNOFF system program.

The 2741 terminal operates in half duplex mode and transmits and receives non-ASCII characters. The terminal echoes locally; the computer does not echo the characters. This action differs greatly from ASCII terminals which operate under RSTS/E in full duplex mode.

Half duplex operation of the 2741 terminal involves stricter intercommunication between the device and the computer. For example, when the user types the RETURN key to enter a line, the keyboard locks until the computer accepts the line and sends an EOT character to unlock the keyboard. This locking and unlocking action is noticeable and may be annoying to the fast typist.

Many graphic code and keyboard arrangements are available for 2741 compatible terminals. RSTS-11 supports the four most common codes: Correspondence, Extended Binary Coded Decimal, Binary Coded Decimal and CALL 360 BASIC. Since the system can be configured for any combination of the four codes, it is advisable to consult the system manager for information concerning which codes are available at the local installation.

2741 terminals do not generate all the characters normally used for time sharing operations under RSTS-11. The system interprets certain keys in special ways described in the following subsections. The four supported keyboard arrangements are shown in Figures 5-13 through 5-16 at the end of the section.

5.8.1 The ATTN Key

The 2741 terminal has an ATTN (Attention) key usually on the upper right hand portion of the keyboard. (Some terminals have an equivalent key called the BREAK key.) The key has

---

[1] This feature is available only on Version 5 (RSTS/E) systems.

[2] The RSTS/E 2741 code has been tested with IBM, DATEL, and TREND DATA terminals. Digital Equipment Corporation makes no commitment to support 2741-type terminals made by other manufacturers.

several uses under RSTS-11 depending upon whether the terminal is transmitting data to or receiving data from the computer.

The terminal is considered transmitting data to the computer when it is at BASIC-PLUS command level or in the program input state. The terminal is receiving characters whenever the system performs output to the device.

When the terminal is transmitting data, the ATTN key can have two effects. If pressed while the SHIFT key is in its upper case position, the key generates the effect of a CTRL/C combination as described in Chapter 3. The system echoes the transmission as either ↑C or ¢C. If pressed while the SHIFT key is in its lower case position, the key generates the effect of a CTRL/U combination (erase line) as described in Chapter 3. The system echoes the lower case ATTN key as either ↑U or ¢U.

When the terminal is receiving data, pressing the ATTN key with the SHIFT key in its upper case or lower case position generates the effect of a CTRL/C combination as described in Chapter 3. This action allows the user to interrupt computer printout and return control to BASIC-PLUS command level.

NOTE

The system reacts to the CTRL/C combination on a 2741 type terminal as if the user typed two such combinations in rapid succession. Programs executing the CTRL/C trap enable system function cannot trap a CTRL/C combination from a 2741 type terminal.

5.8.2 The RETURN Key

The RETURN key generates one of two characters depending upon whether the SHIFT key is in its upper or lower case position. If typed as a lower case character, the key generates a CR character (CHR$(13)). If typed as an upper case character, the key generates a LF character (CHR$(10)). The system thus allows the user to continue BASIC-PLUS statement lines when he enters source code from a 2741 terminal.

5.8.3 The BKSP Key

The BKSP key generates one of two characters depending upon whether the SHIFT key is in its upper or lower case position. If typed as an upper case character, the key backspaces the typing element one character position and generates the BACKSPACE character (CHR$(8)). If typed as a lower case character, the key backspaces the typing element one character position and generates the DEL (RUBOUT) character (CHR$(127)). In the latter case, the system

deletes from memory the last character typed. This action is similar to the RUBOUT key typed at an ASCII terminal except that any replacement character typed is printed over the deleted character.

5.8.4 Bracket Characters

Since most 2741 terminals have no bracket characters, the system accepts (and) characters typed in place of [and] characters. This feature allows users to delimit a project-programmer number with open and close parenthesis characters rather than with bracket characters. For example, (100,100) is equivalent to [100,000]. System programs translate [and] characters to (and) characters before processing commands.

5.8.5 Changing Codes

If the system is configured to handle more than one transmission code, the user can change the code recognized by the system. This feature is available because the system initializes each device to a default code when time sharing operations begin. Therefore, if an individual terminal does not employ the default code, the user must change the code to operate on the system.

To enable the system to recognize codes generated by one of the keyboards shown in Figures 5-13 through 5-16, the user must type the numeral 1 followed by the upper case ATTN key. (See the description of the ATTN key in Section 5.8.1). For example,

```
1¢C
E963
Ready
```

As a result, the system changes the code conversion table for the device. It prints an identifier which associates both the code and its required typing sphere and prints the Ready message. Table 5-8 shows the identifiers and related reference information.

Table 5-8

2741 Transmission Code Identifiers

Identifier	Keyboard Code	Figure	Typing Sphere
C029	Correspondence	5-13	1167-029
E963	EBCD	5-14	1167-963
B938	BCD	5-15	1167-938
C087	CALL/360 BASIC	5-16	1167-087

The code table the system uses and the typing sphere the device uses must match the particular keyboard. If either the typing sphere or the code table is incorrect, the terminal produces garbled output. For example,

1'g
X92#
Nupvi

To verify the typing sphere, the user can compare the typing sphere number shown in Table 5-8 with that stamped on the top edge of the type ball under the retaining clamp lever. To verify the keyboard, compare it with the one shown in the figure referred to in Table 5-8. If the typing sphere does not match the keyboard, install the correct type ball. Otherwise, continue changing codes until the system prints a recognizable identifier and Ready message. If the user changes codes four times and the system does not print a recognizable message, the terminal cannot be used unless the system is regenerated to support that code and keyboard arrangement.

The graphics of the typing spheres for each keyboard are compatible with conventional system graphics except for certain special characters. Where differences exist, the conventional character is shown in the related keyboard layout above the graphic printed by the sphere. All system programs accept lower case characters by performing a conversion to upper case with the CVT$$ function described in Section 12.5 of the BASIC-PLUS Language Manual.

Correspondence Code Keyboard

The following special characters and system actions are produced on this keyboard.

Character or Action	2741 Method
LF (Line Feed)	RETURN key in upper case
CR (RETURN)	RETURN key in lower case
DEL (RUBOUT)	BKSP key in lower case
BKSP (BACKSPACE)	BKSP key in upper case
CTRL/C combination	ATTN key in upper case
CTRL/U combination	ATTN key in lower case
CTRL/O combination	None
CTRL/Z combination	None
Change Code	1 followed by the upper case ATTN key

Figure 5-13 shows the correspondence keyboard layout and specifies the required typing sphere. Special system characters such as ↑, and \, which do not have a graphic equivalent are shown in the area above the key that is used as a replacement. For example, the terminal prints the ¢ character to designate the ↑ character. Abbreviations above or below keys are defined in the legend.

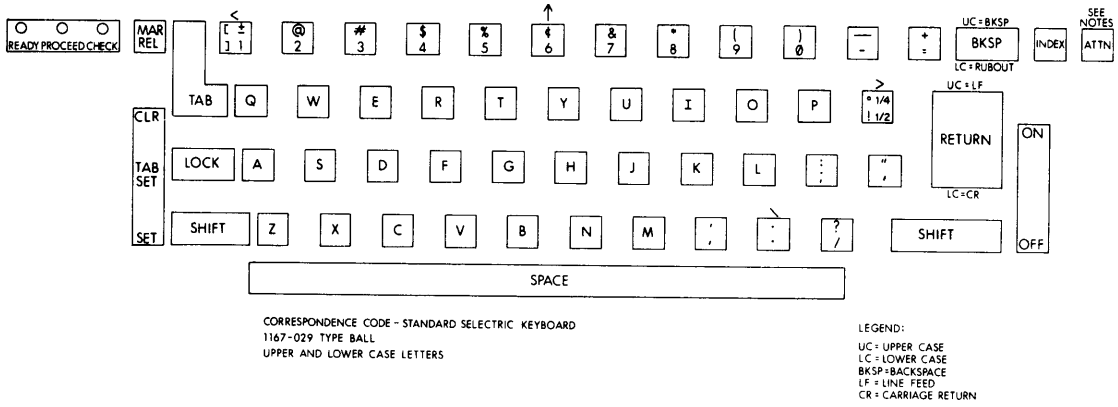


Figure 5-13 Correspondence Code Keyboard

EBCD Keyboard

The following special characters and system actions are produced on this keyboard.

Character or Action	2741 Method
LF (LINE FEED)	RETURN key in upper case
CR (RETURN)	RETURN key in lower case
DEL (RUBOUT)	BKSP key in lower case
BKSP (BACKSPACE)	BKSP key in upper case
CTRL/C combination	ATTN key in upper case
CTRL/U combination	ATTN key in lower case
CTRL/O combination	None
CTRL/Z combination	None
Change Code	1 followed by the upper case ATTN key

Figure 5-14 shows the EBCD keyboard layout and specifies the required typing sphere. Special system characters such as ↑, and \ which do not have a graphic equivalent are shown in the area above the key that is used as a replacement. For example, the terminal prints a ¢ character in place of the ↑ character. Abbreviations above or below keys are defined in the legend.

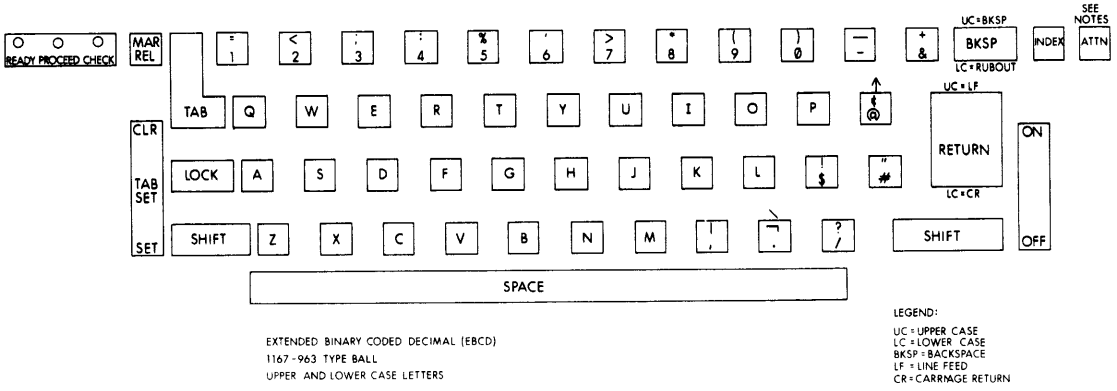


Figure 5-14 EBCD Keyboard

BCD Keyboard

The following special characters and system actions are produced on this keyboard.

Character Action	2741 Method
LF (LINE FEED)	RETURN key in upper case
CR (RETURN)	RETURN key in lower case
DEL (RUBOUT)	BKSP key in lower case
BKSP (BACKSPACE)	BKSP key in upper case
CTRL/C combination	ATTN key in upper case
CTRL/U combination	ATTN key in lower case
CTRL/O combination	None
CTRL/Z combination	None
Change Code	1 followed by the upper case ATTN key

Figure 5-15 shows the BCD keyboard layout, and specifies the required typing sphere. Special system characters such as ↑, and \, which do not have a graphic equivalent are shown in the area above the key that is used as a replacement. For example, the terminal prints the ¢ character to designate the ↑ character. Abbreviations above or below keys are defined in the legend.

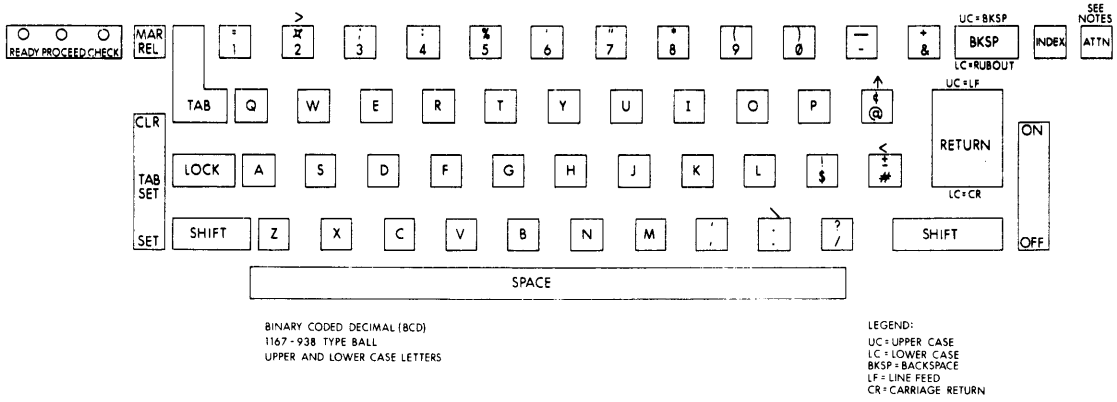


Figure 5-15 BCD Keyboard

CALL/360 BASIC Keyboard

The following special characters and system action are produced on this keyboard.

Character or Action	2741 Method
LF (LINE FEED)	RETURN key in upper case
CR (RETURN)	RETURN key in lower case
DEL (RUBOUT)	BKSP key in lower case
BKSP (BACKSPACE)	BKSP key in upper case
CTRL/C combination	ATTN key in upper case
CTRL/U combination	ATTN key in lower case
CTRL/O combination	None
CTRL/Z combination	None
Change Code	1 followed by the upper case ATTN key

Figure 5-16 shows the CALL/360 BASIC keyboard layout and specifies the required typing sphere. Special system characters such as ↑, \, [, ?, and] which do not have a graphic equivalent are shown in the area above the key that is used as a replacement. For example, the terminal prints a ¢ character to designate the ↑ character. Abbreviations above or below keys are defined in the legend. This keyboard does not produce lower case letters.

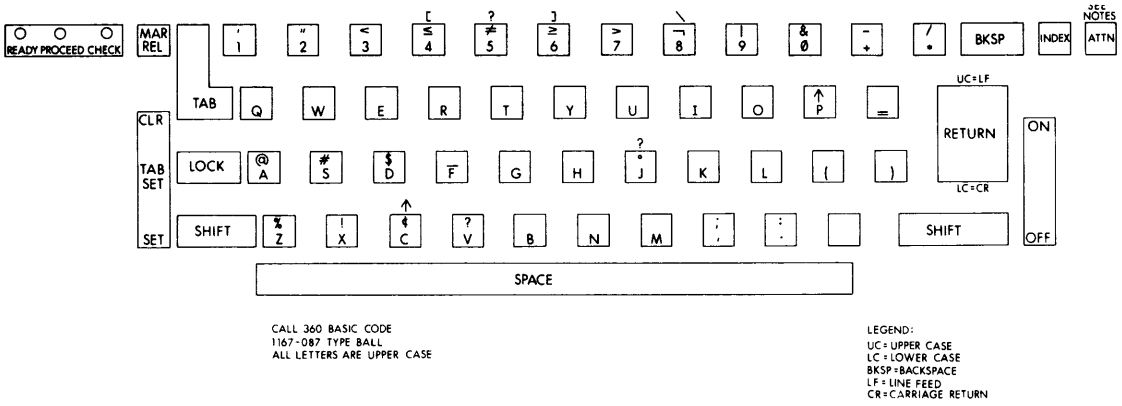


Figure 5-16 CALL/360 BASIC

5.9 LA36 DECWRITER II

DECwriter II offers fast, reliable operation and can be easily interfaced as a remote terminal or local computer I/O device. This terminal prints 30 characters per second and up to 132 characters per line. Characters are formed from a 7 x 7 dot matrix. Character spacing is 10 characters per inch, horizontal and 6 lines per inch, vertical. An original and up to five copies can be printed, and forms can be any width from 3 inches to 15 inches wide.

Table 5-9 describes the purpose of each operator control on the DECwriter II.

Table 5-9
DECwriter II Operator Controls

Control	Meaning
Power On-Off	Applies and removes AC power to entire machine.
Line/Local	Selects on line or local operation.
Baud-rate - 110, 150, 300	3-position switch selects the baud rate clock frequency for communications line operation.
Forms thickness adjustment	Located on right side of print head carriage. Selects proper gap for 1- through 6-part form. Approximately 1 click for each part.
Right Tractor Adjustment	Thumb screw may be loosened to allow movement of right tractor for various forms widths.
Fine Vertical Tractor Release	Line feed knob may be depressed inward and rotated in the approximate direction for precise location of printing with respect to vertical zones.

5.10 RX11 FLOPPY DISK

The RX11 floppy disk system provides a low cost, random access, mass memory device capable of storing up to 256,256 8-bit bytes of data in a non-file structured format. The floppy disk itself is a thin flexible, oxide-coated disk, similar in size to a 45 RPM phonograph record. The disk is recorded on one side and is housed in an 8 inch square flexible envelope. The envelope has a large center hole for the drive spindle, a small hole for track index sensing, and a larger slit for the read/write head.

Each RX11 floppy disk controller handles one or two drives, mounted side-by-side horizontally. The lower numbered unit is on the left; the higher numbered unit on the right. RSTS/E systems support up to four controllers (and eight drives).

Once power is applied to the floppy disk system, raise the door of the desired disk drive by pulling up on the centrally located latch. Then simply insert the floppy disk (still housed in its square envelope) into the drive, label-side up, and read/write head slit first. Close the door firmly. The floppy disk is now ready to use; the disk rotates at its operating speed immediately.

To remove the disk, simply open the door and pull the disk envelope out. Once again, the disk stops rotating as soon as the door is opened.

APPENDIX A

BASIC-PLUS LANGUAGE SUMMARY

All manual section numbers given in this Appendix refer to sections in the BASIC-PLUS Language Manual.

A.1 SUMMARY OF VARIABLE TYPES

Type	Variable Name	Examples
Floating Point	single letter optionally followed by a single digit	A I X3
Integer	any floating point variable name followed by a % character	B% D7%
Character String	any floating point variable name followed by a $ character	M$ R1$
Floating Point Matrix	any floating point variable name followed by one or two dimension elements in parentheses	S(4) E(5,1) N2(8) V8(3,3)
Integer Matrix	any integer variable name followed by one or two dimension elements in parentheses	A%(2) I%(3,5) E3%(4) R2%(2,1)
Character String Matrix	any character string variable name followed by one or two dimension elements in parentheses	C$(1) S$(8,5) A2$(8) V1$(4,2)

A.2 SUMMARY OF OPERATORS

Type	Operator		Operates Upon
Arithmetic	-	unary minus	numeric variables and constants
	↑	exponentiation	
	*,/	multiplication, division	
	+,-	addition, subtraction	
Relational	=	equals	string or numeric variables and constants
	<	less than	
	<=	less than or equal to	
	>	greater than	
	>=	greater than or equal to	
	<>	not equal to	
	==	approximately equal to	numeric variables
	==	exactly equal to	string variables
Logical	NOT	logical negation	integer variables and integer-valued expressions
	AND	logical product	
	OR	logical sum	
	XOR	logical exclusive or	
	IMP	logical implication	
	EQV	logical equivalence	
String	+	concatenation	string constants and variables
Matrix	+,-	addition and subtraction of matrices or equal dimensions, one operator per statement	dimensioned variables. See Section 7.6.1 for further details.
	*	multiplication of conformable matrices	
	*	scalar multiplication of a matrix, see Section 7.5.1.	

A.3 SUMMARY OF FUNCTIONS

Under the Function column, the function is shown as:

Y=function

where the characters % and $ are appended to Y if the value returned is an integer or character string.

A floating value (X), where specified, can always be replaced by an integer value. An integer value (N%) can always be replaced by a floating value (an implied FIX is done) except in the CVT%$ and MAGTAPE functions (the symbol I% is used to indicate the necessity for an integer value).

Section numbers found in the Explanation column refer to sections in the BASIC-PLUS Language Manual.

Type	Function	Explanation
Mathematical	Y=ABS(X)	returns the absolute value of X.
	Y+ATN(X)	returns the arctangent of X, where X is in radians.
	Y=COS(X)	returns the cosine of X, where X is in radians.
	Y=EXP(X)	returns the value of e↑X, where e=2.71828.
	Y=FIX(X)	returns the truncated value of X, SGN(X)*INT(ABS(X))
	Y=INT(X)	returns the greatest integer in X which is less than or equal to X.
	Y=LOG(X)	returns the natural logarithm of X, $\log_e X$.
	Y=LOG10(X)	returns the common logarithm of X, $\log_{10} X$.
	Y=PI	has a constant value of 3.14159.
	Y=RND	returns a random number between 0 and 1.
	Y=RND(X)	returns a random number between 0 and 1.
	Y=SGN(X)	returns the sign function of X, a value of 1 preceded by the sign of X.
	Y=SIN(X)	returns the sine of X, where X is in radians.
	Y=SQR(X)	returns the square root of X.
	Y=TAN(X)	returns the tangent of X, where X is in radians.
Print	Y%=POS(X%)	returns the current position of the print head for I/O channel X, 0 is the user's Teletype. (This value is imaginary for disk files.)
	Y$=TAB(X%)	moves print head to position X in the current print record, or is disregarded if the current position is beyond X. The first position is counted as 0.
String	Y%=ASCII(A$)	returns the ASCII value of the first character in the string A$.
	Y$=CHR$(X%)	returns a character string having the ASCII value of X. Only one character is generated.
	Y$=CVT%$(I%)	maps integer into 2-character string, see Section 12.5.
	Y$=CVTF$(X)	maps floating-point number into 4- or 8-character string, see Section 12.5.
	Y%=CVT$%(A$)	maps first 2 characters of string A$ into an integer, see Section 12.5.
	Y=CVT$F(A$)	maps first 4 or 8 characters of string A$ into a floating-point number. See Section 12.5.

Type	Function	Explanation
String cont'd.	Y$=CVT$$(A$,I%)[1]	converts string A$ to string Y$ according to value of I%. See Section 12.5.
	Y$=STRING$(N1%,N2%)[1]	creates string Y$ of length N1 and characters whose ASCII decimal value is N2. See Section 5.5.
	Y$=LEFT(A$,N%)	returns a substring of the string A$ from the first character to the Nth character (the leftmost N characters).
	Y$=RIGHT(A$,N%)	returns a substring of the string A$ from the Nth to the last character; the rightmost characters of the string starting with the Nth character.
	Y$=MID(A$,N1%,N2%)	returns a substring of the string A$ starting with the N1 and being N2 characters long (the characters between and including the N1 to N1+N2-1 characters).
	Y%=LEN(A$)	returns the number of characters in the string A$, including trailing blanks.
	Y%=INSTR(N1%,A$,B$)	indicates a search for the substring B$ within the string A$ beginning at character position N1. Returns a value Ø if B$ is not in A$, and the character position of B$ if B$ is found to be in A$ (character position is measured from the start of the string).
	Y$=SPACE$(N%)	indicates a string of N spaces, used to insert spaces within a character string.
	Y$=NUM$(N%)	indicates a string of numeric characters representing the value of N as it would be output by a PRINT statement. For example: NUM$(1.ØØØØ) = (space)1(space) and NUM$(-1.ØØØØ) = -1(space).
	Y=VAL(A$)	computes the numeric value of the string of numeric characters A$. If A$ contains any character not acceptable as numeric input with the INPUT statement, an error results. For example: VAL("15")=15
	Y$=XLATE(A$,B$)	translate A$ to the new string Y$ by means of the table string B$, see Section 12.7.

[1]These functions are not available prior to Version 5B(RSTS/E) systems.

Type	Function	Explanation
System	Y$=DATE$(Ø%)	returns the current date in the following format: Ø2-Mar-71
	Y$=DATE$(N%)	returns a character string corresponding to a calendar date as follows: N=(day of year)+[(number of years since 1970)*1000] DATE$(1) = "Ø1-Jan-7Ø" DATE$(125) = "Ø5-May-7Ø" DATE$(4125) = "Ø5-May-74"
	Y$=TIME$(Ø%)	returns the current time of day as a character string as follows: TIME$(Ø) = "Ø5:3Ø PM" or "17:3Ø "
	Y$=TIME$(N%)	returns a string corresponding to the time at N minutes before midnight, for example: TIME$(1) = "11:59 PM" or "23:59 " TIME$(144Ø) = "12:ØØ AM" or "ØØ:ØØ " TIME$(721) = "11:59 AM" or "11:59 "
	Y=TIME(Ø%)	returns the clock time in seconds since midnight, as a floating point number.
	Y=TIME(1%)	returns the central processor time used by the current job in tenths of seconds.
	Y=TIME(2%)	returns the connect time (during which the user is logged into the system) for the current job in minutes.
	Y=TIME(3%)[1]	returns to Y the decimal number of kilo-core ticks (kct's) used by this job. See Section 8.8.
	Y=TIME(4%)[1]	returns to Y the decimal number of minutes of device time used by this job. See Section 8.8.
	Y%=STATUS[1]	returns to Y% the status of a channel as of the most recent OPEN statement executed in the program. See Section 12.3.5.
	Y%=BUFSIZ(N)[1]	returns to Y% the buffer size of the device or file open on Channel N. See Section 12.3.4.
	Y%=LINE	returns to Y% the line number of the statement being executed at the time of an interrupt. See Section 4.5.
	Y%=ERR	returns value associated with the last encountered error if an ON ERROR GOTO statement appears in the program. See Section 8.4.
	Y%=ERL	returns the line number at which the last error occurred if an ON ERROR GOTO statement appears in the program. See Section 8.4.3.
	Y%=SWAP%(N%)	causes a byte swap operation on the two bytes in the integer variable N%.

[1]These functions are not available prior to Version 5B(RSTS/E) systems.

Type	Function	Explanation
	Y$=RAD$(N%)	converts an integer value to a 3-character string and is used to convert from Radix-5Ø format back to ASCII. See Appendix D.
Matrix	MAT Y=TRN(X)	returns the transpose of the matrix X, see Section 7.6.2.
	MAT Y=INV(X)	returns the inverse of the matrix X, see Section 7.6.2.
	Y=DET	following an INV(X) function evaluation, the variable DET is equivalent to the determinant of X.
	Y%=NUM	following input of a matrix, NUM contains the number of rows input, or, in the case of a dimensional matrix, the number of elements entered.
	Y%=NUM2	following input of a matrix, NUM2 contains the number of elements entered in that row.
Input/Output	Y%=RECOUNT	returns the number of characters read following every input operation. Used primarily with non-file structured devices. See Section 12.3.1.

A.4 SUMMARY OF BASIC-PLUS STATEMENTS

The following summary of statements available in the BASIC-PLUS language defines the general format for the statement as a line in a BASIC program. If more detailed information is needed, the reader is referred to the section(s) in the BASIC-PLUS Language Manual dealing with that particular statement.

NOTE: All section numbers refer to the BASIC-PLUS Language Manual.

In these definitions, elements in angle brackets are necessary elements of the statement. Elements in square brackets are necessary elements of which the statement may contain one. Elements in braces are optional elements of the statement.

Where the term line number ({*line number*}) is shown in braces, this statement can be used in immediate mode.

The various elements and their abbreviations are described below:

variable or *var*	Any legal BASIC variable as described in A.1 or Section 2.5.2.
line number	Any legal BASIC line number described in Section 2.2.
expression or *exp*	Any legal BASIC expression as described in Section 2.5.
message	Any combination of characters.
condition or *cond*	Any logical condition as described in Section 3.5.
constant	Any acceptable integer constant (need not contain a % character).
argument(s) or *arg*	Dummy variable names.
statement	Any legal BASIC-PLUS statement.
string	Any legal string constant or variable as described in Section 5.1.
protection	Any legal protection code as described in Section 9.1.
value(s)	Any floating point, integer, or character string constant.
list	The legal list for that particular statement.
dimension(s)	One or two dimensions of a matrix, the maximum dimension(s) for that particular statement.

REM

{line number} REM <*message*> 3.1

{line number}{<*statement*>}!<*message*>

```
10  REM  THIS IS A COMMENT
15  PRINT   !PERFORM A CR/LF
```

LET

{line number}{LET}<*var*>{,<*var*>,<*var*>...} = <*exp*> 3.2

```
55  LET A=40: B=22
60  B,C,A=4.2     !MULTIPLE ASSIGNMENT
```

DIM

line number DIM<*var(dimension(s))*> 3.6.2

```
10  DIM A(20), B$(5,10), C%(45)                    7.1
```

line number DIM #<*constant*>,<*var(dimension(s))*>=<*constant*> 11.1

```
75  DIM #4, A$(100)=32,B(50,50)                    11.1
```

RANDOMIZE 3.7.2

line number RANDOM{IZE}

```
55  RANDOMIZE
70  RANDOM
```

IF-THEN, IF-GOTO

line number IF <*cond*> [THEN<*statement*> / THEN<*line number*> / GOTO<*line number*>] 3.5

```
55  IF A>B OR B>C THEN PRINT "NO"
60  IF FNA(R)= B THEN 250
95  IF L<X↑2 AND L<>0 GOTO 345
```

IF-THEN-ELSE 8.5

line number IF <*cond*> [THEN<*statement*> / THEN<*line number*> / GOTO<*line number*>] { ELSE<*statement*> / ELSE<*line number*> }

```
30  IF B=A THEN PRINT "EQUAL" ELSE PRINT "NOT EQUAL"
50  IF A>N THEN 200 ELSE PRINT A
75  IF B==R THEN STOP ELSE 80
```

FOR

line number FOR <*var*>= <*exp*>TO <*exp*> {STEP<*exp*>} 3.6.1

```
20  FOR I=2 TO 40 STEP 2
55  FOR N=A TO A+R
```

FOR-WHILE, FOR-UNTIL 8.6

line number FOR <*var*> = <*exp*> {STEP<*exp*>}[WHILE / UNTIL] <*cond*>

```
84  FOR I = 1 STEP 3 WHILE I<X
74  FOR N = 2 STEP 4 UNTIL N>A OR N=B
05  FOR B= 1 UNTIL B>10
```

NEXT 3.6.1

line number NEXT <*var*>

```
25  NEXT I
60  NEXT N
```

DEF, single line — 3.7.3, 5.5.1, 6.4

line number DEF FN<*var*>(*arg*) =<*exp*(*arg*)>

```
2Ø  DEF FNA(X,Y,Z)=SQR(X↑2+Y↑2+Z↑2)
```

DEF, multiple line — 8.1

line number DEF FN<*var*>(*arg*)
<*statements*>
line number FN<*var*>=<*exp*>
line number FNEND

```
1Ø  DEF FNF(M)   !FACTORIAL FUNCTION
2Ø  IF M=1 THEN FNF=1 ELSE FNF=M*FNF(M-1)
3Ø  FNEND
```

GOTO — 3.4

line number GOTO <*line number*>

```
1ØØ  GOTO 5Ø
```

ON-GOTO — 8.2

line number ON <*exp*> GOTO <*list of line numbers*>

```
75  ON X GOTO 95, 15Ø, 45, 2ØØ
```

GOSUB — 3.8.1

line number GOSUB <*line number*>

```
9Ø  GOSUB 2ØØ
```

ON-GOSUB — 8.3

line number ON <*exp*> GOSUB <*list of line numbers*>

```
85  ON FNA(M) GOSUB 2ØØ, 25Ø, 4ØØ, 375
```

RETURN — 3.8.2

line number RETURN

```
375  RETURN
```

CHANGE — 5.2

{*line number*} CHANGE [<*array name*> / <*string var*>] TO [<*string var*> / <*array name*>]

```
25  CHANGE A$ TO X
7Ø  CHANGE M TO R$
75  CHANGE B TO B$
```

OPEN — 9.2, 9.2.1, 9.2.2

{*line number*} OPEN<*string*>{FOR [INPUT / OUTPUT]}AS FILE <*exp*>
{,RECORDSIZE<*exp*>}{,CLUSTERSIZE <*exp*>}{,MODE <*exp*>}

```
1Ø  OPEN "PP:" FOR OUTPUT AS FILE B1
2Ø  OPEN "FOO" AS FILE 3
3Ø  OPEN "DT4:DATA.TR" FOR INPUT AS FILE 1Ø
```

CLOSE — 9.3

{*line number*} CLOSE <*list of exp*>

```
1ØØ  CLOSE 2
255  CLOSE 1Ø, 4, N1
```

READ — 3.3.1, 5.3, 6.3, 10.1

line number READ <*list of variables*>

```
25  READ A, B$, C%, F1, R2, B(25)
```

DATA — 3.3.1, 5.3, 6.3

```
line number  DATA <list of values>
      3ØØ    DATA 4.3, "STRING",85,49,75.Ø4,1Ø
```

RESTORE — 3.3.1, 10.2

```
line number  RESTORE
       125   RESTORE
```

PRINT — 3.3.2, 5.4, 6.3, 10.3, 10.3.1, 10.3.2

```
{line number} PRINT{{#<exp>,}<list>}
        25    PRINT  !GENERATES  CR/LF
        75    PRINT "BEGINNING OF OUTPUT";I,A*I
        45    PRINT #4,"OUTPUT TO DEVICE"FNM(A)↑2;B;A
```

PRINT USING — 10.3.3

```
{line number} PRINT {#<exp>,}USING <string>, <list>
        54    PRINT USING "##.##",A
        55    PRINT #3, USING"\\###.##   \\##↑↑↑↑","A=",A,"B=",B
        56    PRINT #7, USING B$,A,B,C
```

INPUT — 3.3.3, 5.3, 10.4, 10.4.1

```
{line number} INPUT {#<exp>,}<list>
        25    INPUT "TYPE YOUR NAME ",A$
        55    INPUT #8, A, N, B$
```

INPUT LINE — 5.3

```
{line number} INPUT LINE {#<exp>,} <string>
        4Ø    INPUT LINE  R$
        75    INPUT LINE  #1, E$
```

NAME-AS — 9.4

```
{line number} NAME <string> AS <string>
       455    NAME "NONAME" AS "FILE1<48>"
       27Ø    NAME "DT4:MATRIX" AS "MATA1<48>"
```

KILL — 9.5

```
{line number} KILL <string>
        45    KILL "NONAME"
```

ON ERROR GOTO — 8.4

```
line number  ON ERROR GOTO {<line number>}
        1Ø   ON ERROR GOTO 5ØØ
       525   ON ERROR GOTO !DISABLES ERROR ROUTINE
       526   ON ERROR GOTO Ø  !DISABLES ERROR ROUTINE
```

RESUME — 8.4.1

```
line number  RESUME {<line number>}
      1ØØØ   RESUME    !OR RESUME Ø ARE EQUIVALENT
       655   RESUME  2ØØ
```

CHAIN — 9.6

```
line number   CHAIN <string> {<exp>}
       375    CHAIN "PROG2"
       5ØØ    CHAIN "PROG3" 75
       6ØØ    CHAIN "PROG3" A
```

STOP 3.9

```
line number   STOP
        75    STOP
```

END 3.9

```
line number   END
       545    END
```

Matrix Statements

MAT READ 7.2

```
 line number  MAT READ <list of matrices>
        55    DIM A(2Ø), B$(32), C%(15,1Ø)
        9Ø    MAT READ A, B$(25), C%
```

MAT PRINT 7.3

```
{line number} MAT PRINT{#<exp>,} <matrix name>
        1Ø    DIM A(2Ø), B(15,2Ø)
        9Ø    MAT PRINT A;          !PRINT 1Ø*1Ø MATRIX, PACKED
        95    MAT PRINT B(1Ø,5),    !PRINT 1Ø*5 MATRIX, FIVE
                                    !ELEMENTS PER LINE
        97    MAT PRINT #2, A;      !PRINT ON OUTPUT CHANNEL 2
```

MAT INPUT 7.4

```
{line number} MAT INPUT{#<exp>,} <list of matrices>
        1Ø    DIM B$(4Ø), F1%(35)
        2Ø    OPEN "DT3:FOO" FOR INPUT AS FILE 3
        3Ø    MAT INPUT #3, B4, F1%
```

MAT Initialization 7.5

{*line number*} MAT <*matrix name*>= [ZER / CON / IDN] {*dimension(s)*}

```
        1Ø    DIM B(15,1Ø), A(1Ø), C%(5)
        15    MAT C% = CON          !ALL ELEMENTS OF C%(I)=1
        2Ø    MAT B = IDN(1Ø,1Ø)    !IDENTITY MATRIX 1Ø*1Ø
        95    MAT B = ZER(N,M)      !CLEARS AN N BY M MATRIX
```

Statement Modifiers (can be used in immediate mode)

IF 8.7.1

```
<statement> IF  <condition>
        1Ø    PRINT X IF X<>Ø
```

UNLESS 8.7.2

```
<statement> UNLESS <condition>
        45    PRINT A UNLESS A=Ø
```

FOR 8.7.3

```
<statement> FOR <var> = <exp> TO <exp>{STEP<exp>}
        75    LET B$(I) = "PDP-11" FOR I = 1 TO 25
        8Ø    READ A(I) FOR I=2 TO 8 STEP 2
```

WHILE 8.7.4

```
<statement> WHILE <condition>
        1Ø    LET A(I) = FNX(I) WHILE I<45.5
```

BASIC-PLUS Language Manual Section

UNTIL 8.7.5

```
<statement> UNTIL <condition>
        115  IF B Ø THEN A(I)=B UNTIL I>5
```

System statements

```
<line number> SLEEP  <expression>                                  8.8
        1ØØ  SLEEP(2Ø)  !DISMISS JOB FOR 2Ø SEC.

<line number> WAIT  <expression>                                   8.8
        525  WAIT(A%+5)  !WAIT A%+5 SEC. FOR INPUT
```

Record I/O Statements

```
<line number> LSET<string var>{,<string var>}=<string>            12.4.3
         9Ø  LSET B$="XYZ"

<line number> RSET<string var>{,<string var>}=<string>            12.4.3
         25Ø  RSET C$="6789Ø"

<line number> FIELD#<expr>,<expr>AS<string var>{,<expr>AS<string var>}
         75  FIELD#2%,1Ø% AS A$, 2Ø% AS B$                        12.4.2

<line number> GET#<expr>{,RECORD<expr>}                            12.3
        1ØØ  GET#1%,RECORD 99%

<line number> PUT#<expr>{,RECORD<expr>}{,COUNT<expr>}              12.3
        5ØØ  PUT#1%,COUNT 8Ø%

<line number> UNLOCK#<expr>                                        10.5.1
        7ØØ  UNLOCK #3%
```

APPENDIX B

BASIC-PLUS COMMAND SUMMARY

Command	Explanation	Section in RSTS-11 System User's Guide
APPEND	Used to include contents of a previously saved source program in current program.	2.4.3
ASSIGN	Used to reserve an I/O device for the use of the individual issuing the command. The specified device can then be given commands only from the terminal which issued the ASSIGN. Also establishes a logical name for a device, establishes an account for the @ character, and assigns a default protection code.	2.6.3 2.7
ATTACH	Attaches a detached job to the current terminal.	4.1
BYE	Indicates to RSTS that a user wishes to leave the terminal. Closes and saves any files remaining open for that user.	2.1.3
CAT CATALOG	Returns the user's file directory. Unless another device is specified following the term CAT or CATALOG, the disk is the assumed device.	2.5.2
CCONT	For privileged users. Same as CONT command but detaches job from terminal.	2.2.9
COMPILE	Allows the user to store a compiled version of his BASIC program. The file is stored on disk with the current name and the extension .BAC. Or, a new file name can be indicated and the extension .BAC will still be appended.	2.3.3
CONT	Allows the user to continue execution of the program currently in core following the execution of a STOP statement.	2.2.8
DEASSIGN	Used to release the specified device for use by others. If no particular device is specified, all devices assigned to that terminal are released. An automatic DEASSIGN is performed when the BYE command is given. Also releases any logical name for a device.	2.6.4 2.7
DELETE	Allows the user to remove one or more lines from the program currently in core. Following the word DELETE the user types the line number of the single line to be deleted or two line numbers separated by a dash (-) indicating the first and last line of the section of code to be removed. Several single lines or line sections can be indicated by separating the line numbers, or line number pairs, with a comma.	2.2.5

Command	Explanation	Section in RSTS-11 System User's Guide
HELLO	Indicates to RSTS that a user wishes to log onto the system. Allows the user to input project-programmer number and password. Also attaches a detached job to the current terminal or changes accounts without having to log off the system.	2.1.2 4.1
KEY	Used to re-enable the echo feature on the user terminal following the issue of a TAPE command. Enter with LINE FEED or ESCAPE key.	2.6.2
LENGTH	Returns the length of the user's current program in core, in 1K increments.	2.5.1
LIST	Allows the user to obtain printed listing at the user terminal of the program currently in core, or one or more lines of that program. The word LIST by itself will cause the listing of the entire user program. LIST followed by one line number will list that line; and LIST followed by two line numbers separated by a dash (-) will list the lines between and including the lines indicated. Several single lines or line sections can be indicated by separating the line numbers, or line number pairs, with a comma.	2.2.4
LISTNH	Same as LIST, but does not print header containing the program name and current date.	2.2.4
LOGIN	Same as HELLO	4.1
NEW	Clears the user's area in core and allows the user to input a new program from the terminal. A program name can be indicated following the word NEW or when the system requests it.	2.2.1
OLD	Clears the user's area in core and allows the user to recall a saved program from a storage device. The user can indicate a program name following the word OLD or when the system requests it. If no device name is given, the file is assumed to be on the system disk. A device specification without a filename will cause a program to be read from an input-only device (such as high-speed reader, card reader).	2.4.2
REASSIGN	Transfers control of a device to another job.	2.6.5
RENAME	Causes the name of the program currently in core to be changed to the name specified after the word RENAME.	2.2.6
REPLACE	Same as SAVE, but allows the user to substitute a new program with the same name for an old program, erasing the old program.	2.4.7

Command	Explanation	Section in RSTS-11 System User's Guide
RUN	Allows the user to begin execution of the program currently in core. The word RUN can be followed by a file name in which case the file is loaded from the system disk, compiled, and run alternatively, the device and file name can be indicated if the file is not on the system disk. A device specification without a file name will cause a program to be read from an input only device (such as high-speed reader, card reader).	2.3.1
RUNNH	Causes execution of the program currently in memory but header information containing the program name and current date is not printed. If a filename is used, the command is executed as if no filename were given.	2.3.1
SAVE	Causes the program currently in core to be saved on the system disk under its current file name with the extension .BAS. Where the word SAVE is followed by a file name or a device and a file name, the program in core is saved under the name given and on the device specified. A device specification without a file name will cause the program to be output to any output only device (line printer, high-speed punch).	2.4.1
SCALE	Sets the scale factor to a designated value or prints the value(s) currently in effect if no value is designated.	2.8
TAPE	Used to disable the echo feature on the user terminal while reading paper tape via the low-speed reader.	2.6.1
UNSAVE	The word UNSAVE is followed by the file name and, optionally, the extension of the file to be removed. The UNSAVE command cannot remove files without an extension. If no extension is specified, the source (.BAS) file is deleted. If no device is specified, the disk is assumed.	2.4.6

Special Control Character Summary

CTRL/C	Causes the system to return to BASIC command mode to allow for issuing of further commands or editing. Echoes on terminal as C.	3.5
CTRL/O	Used as a switch to suppress/enable output of a program on the user terminal. Echoes as ↑O.	3.7
CTRL/Q	When generated by a device on which a CTRL/S has interrupted output, causes computer to resume output at the next character.	3.10

Command	Explanation	Section in RSTS-11 System User's Guide
CTRL/S	When generated by a device for which SCOPE characteristics are set, interrupts computer output on the device until either CTRL/Q or another character is generated.	3.10
CTRL/U	Deletes the current typed line, echoes as ↑U and performs a carriage return/line feed.	3.6
CTRL/Z	Used as an end-of-file character.	3.9
ESCape or ALT MODE Key	Enters a typed line to the system, echoes on the user terminal as a $ character and does not cause a carriage return/line feed.	3.2
LINE FEED Key	Used to continue the current logical line on an additional physical line. Performs a carriage return/line feed operation.	3.3
RETURN Key	Enters a typed line to the system, results in a carriage return/line feed operation at the user terminal.	3.1
RUBOUT Key	Deletes the last character typed on that physical line. Erased characters are shown on the teleprinter between back slashes.	3.4
TAB or CTRL/I	Performs a tabulation to the next of nine tab stops (eight spaces apart) which form the terminal printing line.	3.8
CTRL/L	Generates FORM FEED character and results in four line feed operations at the user terminal.	

APPENDIX C

ERROR MESSAGE SUMMARY

Wherever possible, RSTS follows an error message with the phrase

AT LINE xxxx

where xxxx is the line number of the statement which caused the error. For example:

```
1Ø  TALK
ILLEGAL VERB AT LINE 1Ø
READY
```

The additional message is not printed when no line number can be associated with the error.

```
TALK

WHAT?

READY
```

An (SPR) in the description of any error message in this Appendix indicates an error which should never be seen by a user. If such a message is received, the user should document how he obtained the error and file a Software Performance Report with DEC, including the pertinent output.

C.1 USER RECOVERABLE ERRORS

A (C) in the description of the error message indicates that program execution continues, following printing of the error message, if an ON ERROR GOTO statement is not present. Normally, execution terminates on an error condition, the error message is printed, and the system prints READY. The ERR column gives the value of the ERR variable (see Section 8.4 in the BASIC-PLUS Language Manual).

ERR	Message Printed	Meaning
1	BAD DIRECTORY FOR DEVICE	The directory of the device referenced is an unreadable format.

ERR	Message Printed	Meaning
2	ILLEGAL FILE NAME	The filename specified is not acceptable. It contains unacceptable characters or the filename specification format has been violated.
3	ACCOUNT OR DEVICE IN USE	Removal or dismounting of the account or device cannot be done since one or more users are currently using it.
4	NO ROOM FOR USER ON DEVICE	Storage space allowed for the current user on the device specified has been used or the device as a whole is too full to accept further data.
5	CAN'T FIND FILE OR ACCOUNT	The file or account number specified was not found on the device specified.
6	NOT A VALID DEVICE	Attempt to use an illegal or nonexistent device specification.
7	I/O CHANNEL ALREADY OPEN	An attempt was made to open one of the twelve I/O channels which had already been opened by the program. (SPR)
8	DEVICE NOT AVAILABLE	The device requested is currently reserved by another user.
9	I/O CHANNEL NOT OPEN	Attempt to perform I/O on one of the twelve channels which has not been previously opened in the program.
10	PROTECTION VIOLATION	The user was prohibited from performing the requested operation because the kind of operation was illegal (such as input from a line printer) or because the user did not have the privileges necessary (such as deleting a protected file).
11	END OF FILE ON DEVICE	Attempt to perform input beyond the end of a data file; or a BASIC source file is called into memory and is found to contain no END statement.
12	FATAL SYSTEM I/O FAILURE	An I/O error has occurred on the system level. The user has no guarantee that the last operation has been performed. (SPR)
13	USER DATA ERROR ON DEVICE	One or more characters may have been transmitted incorrectly due to a parity error, bad punch combination on a card, or similar error.
14	DEVICE HUNG OR WRITE LOCKED	User should check hardware condition of device requested. Possible causes of this error include a line printer out of paper or high-speed reader being off-line.
15	KEYBOARD WAIT EXHAUSTED	Time requested by WAIT statement has been exhausted with no input received from the specified keyboard.

ERR	Message Printed	Meaning
16	NAME OR ACCOUNT NOW EXISTS	An attempt was made to rename a file with the name of a file which already exists, or an attempt was made by the system manager to insert an account number which is already within the system.
17	TOO MANY OPEN FILES ON UNIT	Only one open DECtape output file is permitted per DECtape drive. Only one open file per magtape drive is permitted.
18	ILLEGAL SYS() USAGE	Illegal use of the SYS system function.
19	DISK BLOCK IS INTERLOCKED	The requested disk block segment is already in use (locked) by some other user.
20	PACK IDS DON'T MATCH	The identification code for the specified disk pack does not match the identification code already on the pack.
21	DISK PACK IS NOT MOUNTED	No disk pack is mounted on the specified disk drive.
22	DISK PACK IS LOCKED OUT	The disk pack specified is mounted but temporarily disabled.
23	ILLEGAL CLUSTER SIZE	The specified cluster size is unacceptable.
24	DISK PACK IS PRIVATE	The current user does not have access to the specified private disk pack.
25	DISK PACK NEEDS 'CLEANING'	Non-fatal disk mounting error; use the CLEAN operation in UTILTY.
26	FATAL DISK PACK MOUNT ERROR	Fatal disk mounting error. Disk cannot be successfully mounted.
27	I/O TO DETACHED KEYBOARD	I/O was attempted to a hung up dataset or to the previous, but now detached, console keyboard for the job.
28	PROGRAMMABLE ↑C TRAP	ON ERROR-GOTO subroutine was entered through a program trapped CTRL/C. See a description of the SYS system function.
29	CORRUPTED FILE STRUCTURE	Fatal error in CLEAN operation.

30	DEVICE NOT FILE STRUCTURED	An attempt is made to access a device, other than a disk, DECtape, or magtape device, as a file-structured device. This error occurs, for example, when the user attempts to gain a directory listing of a non-directory device.
31	ILLEGAL BYTE COUNT FOR I/O	The buffer size specified in the RECORDSIZE option of the OPEN statement or in the COUNT option of the PUT statement is not a multiple of the block size of the device being used for I/O.
32	NO ROOM AVAILABLE FOR FCB	When the user accesses a file under programmed control in RSTS-11, a system control structure called an FCB requires one small buffer and one small buffer is not available for the FCB.
33	UNIBUS TIMEOUT FATAL TRAP	This hardware error occurs when an attempt is made to address nonexistent memory or an odd address using the PEEK function. An occurrence of this error message in any other case is cause for an SPR.
34	RESERVED INSTRUCTION TRAP	An attempt is made to execute an illegal or reserved instruction or an FPP instruction when floating point hardware is not available. (SPR)
35	MEMORY MANAGEMENT VIOLATION	This hardware error occurs when an illegal Monitor address is specified using the PEEK function. Generation of the error message in situations other than using PEEK is cause for an SPR.
36	SP(R6) STACK OVERFLOW	An attempt to extend the hardware stack beyond its legal size is encountered. (SPR)
37	DISK ERROR DURING SWAP	A hardware error occurs when a user's job is swapped into or out of memory. The contents of the user's job area are lost but the job remains logged into the system and is reinitialized to run the NONAME program. (SPR)
38	MEMORY PARITY FAILURE	A parity error was detected in the memory occupied by this job.
39	MAGTAPE SELECT ERROR	When access to a magtape drive was attempted, the selected unit was found to be off line.
40	MAGTAPE RECORD LENGTH ERROR	When performing input from magtape, the record on magtape was found to be longer than the buffer designated to handle the record.
41	NO RUN-TIME SYSTEM	Reserved.
42	VIRTUAL BUFFER TOO LARGE	Virtual core buffers must be 512 bytes long.

ERR	Message Printed	Meaning
43	VIRTUAL ARRAY NOT ON DISK	A non-disk device is open on the channel upon which the virtual array is referenced.
44	MATRIX OR ARRAY TOO BIG	In-core array size is too large
45	VIRTUAL ARRAY NOT YET OPEN	An attempt was made to use a virtual array before opening the corresponding disk file.
46	ILLEGAL I/O CHANNEL	Attempt was made to open a file on an I/O channel outside the range of the integer numbers 1 to 12.
47	LINE TOO LONG	Attempt to input a line longer than 255 characters (which includes any line terminator). Buffer overflows/
48	FLOATING POINT ERROR	Attempt to use a computed floating point number outside the range $\|1E-38\| \leq n \leq \|1E38\|$ excluding zero. If no transfer to an error handling routine is made, zero is returned as the floating point value. (C)
49	ARGUMENT TOO LARGE IN EXP	Acceptable arguments are within the approximate range $-89 \leq arg \leq +88$. The value returned is zero. (C)
50	DATA FORMAT ERROR	A READ or INPUT statement detected data in an illegal format. For example, 1..3 is an improperly formed number, 1.2 is an improperly formed integer, and X" is an illegal string. (C)
51	INTEGER ERROR	Attempt to use a computed integer outside the range $-32767 \leq n \leq 32767$. If no transfer to an error handling routine is made, zero is returned as the integer value. (C)
52	ILLEGAL NUMBER	Integer or floating point overflow or underflow.
53	ILLEGAL ARGUMENT IN LOG	Negative or zero argument to log function. Value returned is the argument as passed to the function. (C)
54	IMAGINARY SQUARE ROOTS	Attempt to take square root of a number less than zero. The value returned is the square root of the absolute value of the argument. (C)
55	SUBSCRIPT OUT OF RANGE	Attempt to reference an array element beyond the number of elements created for the array when it was dimensioned.
56	CAN'T INVERT MATRIX	Attempt to invert a singular or nearly singular matrix.
57	OUT OF DATA	The DATA list was exhausted and a READ requested additional data.
58	ON STATEMENT OUT OF RANGE	The index value in an ON-GOTO or ON-GOSUB statement is less than one or greater than the number of line numbers in the list.

ERR	Message Printed	Meaning
59	NOT ENOUGH DATA IN RECORD	An INPUT statement did not find enough data in one line to satisfy all the specified variables.
60	INTEGER OVERFLOW, FOR LOOP	The integer index in a FOR loop attempted to go beyond 32766 or below -32766.
61	DIVISION BY Ø	Attempt by the user program to divide some quantity by zero. If no transfer is made to an error handler routine, a Ø is returned as the result. (C)

C.2 NON-RECOVERABLE ERRORS

Message Printed	Meaning
ARGUMENTS DON'T MATCH	Arguments in a function call do not match, in number or in type, the arguments defined for the function.
BAD LINE NUMBER PAIR	Line numbers specified in a LIST or DELETE command were formatted incorrectly.
BAD NUMBER IN PRINT-USING	Format specified in the PRINT-USING string cannot be used to print one or more values.
CAN'T COMPILE STATEMENT	
CAN'T CONTINUE	Program was stopped or ended at a spot from which execution cannot be resumed.
CATASTROPHIC ERROR	The user program data structures are destroyed. This normally indicates a BASIC-PLUS malfunction and, if reproducible, should be reported to DEC on a Software Performance Report form. (SPR)
DATA TYPE ERROR	Incorrect usage of floating-point, integer, or character string format variable or constant where some other data type was necessary.
DEF WITHOUT FNEND	A second DEF statement was encountered in the processing of a user function without an FNEND statement terminating the first user function definition.
END OF STATEMENT NOT SEEN	Statement contains too many elements to be processed correctly.
EXECUTE ONLY FILE	Attempt was made to add, delete or list a statement in a compiled (.BAC) format file.

Message Printed	Meaning
EXPRESSION TOO COMPLICATED	This error usually occurs when parentheses have been nested too deeply. The depth allowable is dependent on the individual expression.
FIELD OVERFLOWS BUFFER	Attempt to use FIELD to allocate more space than exists in the specified buffer.
FILE EXISTS-RENAME/REPLACE	A file of the name specified in a SAVE command already exists. In order to save the current program under the name specified, use the REPLACE command.
FNEND WITHOUT DEF	An FNEND statement was encountered in the user program without a previous DEF statement being seen.
FNEND WITHOUT FUNCTION CALL	A FNEND statement was encountered in the user program without a previous function call having been executed. Function has been placed incorrectly among executable statements or an extra FNEND statement has been found.
FOR WITHOUT NEXT	A FOR statement was encountered in the user program without a corresponding NEXT statement to terminate the loop.
ILLEGAL CONDITIONAL CLAUSE	Incorrectly formatted condition expression.
ILLEGAL DEF NESTING	The range of one function definition crosses the range of another function definition.
ILLEGAL DUMMY VARIABLE	One of the variables in the dummy variable list of a user-defined function is not a legal variable name.
ILLEGAL EXPRESSION	Double operators, missing operators, mismatched parentheses, or some similar error has been found in an expression.
ILLEGAL FIELD VARIABLE	The FIELD variable specified is unacceptable.
ILLEGAL FN REDEFINITION	Attempt was made to redefine a user function.
ILLEGAL FUNCTION NAME	Attempt was made to define a function with a function name not subscribing to the established format.

Message Printed	Meaning
ILLEGAL IF STATEMENT	Incorrectly formatted IF statement.
ILLEGAL IN IMMEDIATE MODE	User issued a statement for execution in immediate mode which can only be performed as part of a program.
ILLEGAL LINE NUMBER(S)	Line number reference outside the range $1 \leq n \leq 32767$.
ILLEGAL MAGTAPE() USAGE	Improper use of the MAGTAPE function.
ILLEGAL MODE MIXING	String and numeric operations cannot be mixed.
ILLEGAL STATEMENT	Attempt was made to execute a statement that did not compile without errors.
ILLEGAL SYMBOL	An unrecognizable character was encountered. For example, a line consisting of a # character.
ILLEGAL VERB	The BASIC verb portion of the statement cannot be recognized.
INCONSISTENT FUNCTION USAGE	A function is being redefined in a manner inconsistent in the number or type of arguments with one or more calls to that function existing in the program.
INCONSISTENT SUBSCRIPT USE	A subscripted variable is being used with a different number of dimensions from the number with which it was originally defined.
K OF CORE USED	Message printed by LENGTH command, preceded by the appropriate number describing the user program currently in core to the nearest 1K.
LITERAL STRING NEEDED	A variable name was used where a numeric or character string was necessary.
MATRIX DIMENSION ERROR	Attempt was made to dimension a matrix to more than two dimensions, or an error was made in the syntax of a DIM statement.
MATRIX OR ARRAY WITHOUT DIM	A matrix or array element was referenced beyond the range of an implicitly dimensioned matrix.
MAXIMUM CORE EXCEEDED	User program grew to be too large to run or compile in the area of core assigned to each user at the given installation.
MISSING SPECIAL FEATURE	User program employs a BASIC-PLUS feature not present on the given installation.

Message Printed	Meaning
MODIFIER ERROR	Attempt to use one of the statement modifiers (FOR, WHILE, UNTIL, IF, or UNLESS) incorrectly.
NEXT WITHOUT FOR	A NEXT statement was encountered in the user program without a previous FOR statement having been seen.
NO LOGINS	Message printed if the system is full and cannot accept additional users or if further logins are disabled by the system manager.
NOT A RANDOM ACCESS DEVICE	Attempt to perform random access I/O to a non-random access device.
NOT ENOUGH AVAILABLE CORE	The already compiled user program is too large to run in the area of core assigned to each user at the given installation.
NUMBER IS NEEDED	A character string or variable name was used where a number was necessary.
1 OR 2 DIMENSIONS ONLY	Attempt was made to dimension a matrix to more than two dimensions.
ON STATEMENT NEEDS GOTO	A statement beginning with ON does not contain a GOTO or GOSUB clause.
PLEASE SAY HELLO	User not logged into the system has typed something other than a legal, logged-out command to the system.
PLEASE USE THE RUN COMMAND	A transfer of control (as in a GOTO, GOSUB or IF-GOTO statement) cannot be performed from immediate mode.
PRINT-USING BUFFER OVERFLOW	Format specified contains a field too large to be manipulated by the PRINT-USING statement.
PRINT-USING FORMAT ERROR	An error was made in the construction of the string used to supply the output format in a PRINT-USING statement.
PROGRAM LOST-SORRY	A fatal system error has occurred which caused the user program to be lost.
REDIMENSIONED ARRAY	Usage of an array or matrix within the user program has caused BASIC-PLUS to redimension the array implicitly.
RESUME AND NO ERROR	A RESUME statement was encountered where no error had occurred to cause a transfer into an error handling routine via the ON ERROR-GOTO statement.

Message Printed	Meaning
RETURN WITHOUT GOSUB	RETURN statement encountered in the user program without a previous GOSUB statement having been executed.
SCALE FACTOR INTERLOCK	An attempt was made to execute a program or source statement with the current scale factor. The program executes but the system uses the scale factor of the program in memory. Use REPLACE and OLD or recompile the program to execute with the current scale factor.
STATEMENT NOT FOUND	Reference is made within the program to a line number which is not within the program.
STOP	STOP statement was executed. The user can usually continue program execution by typing CONT and the RETURN key.
STRING IS NEEDED	A number or variable name was used where a character string was necessary.
SYNTAX ERROR	BASIC-PLUS statement was incorrectly formatted.
TEXT TRUNCATED	No BASIC-PLUS statement can be more than 255 characters long.
TOO FEW ARGUMENTS	The function has been called with a number of arguments not equal to the number defined for the function.
TOO MANY ARGUMENTS	A user-defined function may have up to five arguments.
UNDEFINED FUNCTION CALLED	BASIC-PLUS interpreted some statement component as a function call for which there is no defined function (system or user).
WHAT?	Command or immediate mode statement entered to BASIC-PLUS could not be processed. Illegal verb or improper format error most likely.
WRONG MATH PACKAGE	Program was compiled with an incompatible version of RSTS. Program source must be recompiled.

C.3 SYSTEM IDENTIFICATION MESSAGE

ERR code Ø is associated with the system installation name for use by the system programs.

APPENDIX D

BASIC-PLUS CHARACTER SET

D.1 BASIC-PLUS CHARACTER SET

User program statements are composed of individual characters. Allowable characters come from the following character set:

A through Z	Space
Ø through 9	Tab

and the following special symbols and keys:

Key	Use and Section in BASIC-PLUS Language Manual
$	Used in specifying string variables (Section 5.1), or as the System Library file designator (RSTS-11 System User's Guide).
%	Used in specifying integer variables (Section 6.1). Also denotes account [1,4] (Section 9.1.1).
' "	Used to delimit string constants, i.e., text strings (Section 5.1).
!	Begins comment part of a line (Section 3.1). Also denotes account [1,3] (Section 9.1.1).
:	Separates multiple statements on one line (Section 2.3.1).
\	Separates multiple statements on one line as the colon(:) also does.
#	Denotes a device or file # name, or is used as an output format effector (Chapter 7 and Section 10.3.4). Also denotes account number using current project number with a programmer number of Ø (Section 9.1.1).
,	Output format effector and list terminator (Section 3.3).
;	Output format effector (Section 3.3).
&	Denotes account [1,5] (Section 9.1.1).
@	Denotes the assignable account (Section 9.1.1).
LINE FEED	When used at the end of a line, indicates that the current statement is continued on the next line (Section 2.3.2).
()	Used to group arguments in an arithmetic expression (Section 2.5). Also may be used to group project-programmer number.
[]	Used to group project-programmer number.
<>	Used to delimit file protection codes.
+ - * / ↑	Arithmetic operators (Section 2.5.3).
=	Replacement operator (Section 3.2). Logical equivalence operator (Section 2.5.4).
<	Logical "less than" operator (Section 2.5.4).
>	Logical "greater than" operator (Section 2.5.4).
==	Logical "approximately equal to" operator (Section 2.5.4).

D.2 ASCII CHARACTER CODES

Decimal Value	ASCII Character	RSTS Usage
Ø	NUL	FILL character
1	SOH	
2	STX	
3	ETX	CTRL/C
4	EOT	
5	ENQ	
6	ACK	
7	BEL	BELL
8	BS	
9	HT	HORIZONTAL TAB
1Ø	LF	LINE FEED
11	VT	VERTICAL TAB
12	FF	FORM FEED
13	CR	CARRIAGE RETURN
14	SO	
15	SI	CTRL/O
16	DLE	
17	DC1	
18	DC2	
19	DC3	
2Ø	DC4	
21	NAK	CTRL/U
22	SYN	
23	ETB	
24	CAN	
25	EM	
26	SUB	CTRL/Z
27	ESC	ESCAPE[1]
28	FS	
29	GS	
3Ø	RS	
31	US	
32	SP	SPACE
33	!	
34	"	
35	#	
36	$	
37	%	
38	&	
39	'	APOSTROPHE
4Ø	(	
41	)	
42	*	
43	+	
44	,	COMMA
45	-	
46	.	
47	/	
48	Ø	
49	1	
5Ø	2	
51	3	
52	4	
53	5	
54	6	
55	7	
56	8	
57	9	
58	:	
59	;	
6Ø	<	
61	=	
62	>	
63	?	
64	@	
65	A	
66	B	
67	C	
68	D	
69	E	
7Ø	F	
71	G	
72	H	
73	I	
74	J	
75	K	
76	L	
77	M	
78	N	
79	O	
8Ø	P	
81	Q	
82	R	
83	S	
84	T	
85	U	
86	V	
87	W	
88	X	
89	Y	
9Ø	Z	
91	[	
92	\	Backslash
93	]	
94	^	or ↑
95	_	or ←
96	`	Grave accent
97	a	
98	b	
99	c	
1ØØ	d	
1Ø1	e	
1Ø2	f	
1Ø3	g	
1Ø4	h	
1Ø5	i	
1Ø6	j	
1Ø7	k	
1Ø8	l	
1Ø9	m	
11Ø	n	
111	o	
112	p	
113	q	
114	r	
115	s	
116	t	
117	u	
118	v	
119	w	
12Ø	x	
121	y	
122	z	
123	{	
124	|	Vertical Line
125	}	
126	~	Tilde
127	DEL	RUBOUT

[1]ALTMODE (ASCII 125) or PREFIX (ASCII 126) keys which appear on some terminals are translated internally into ESCAPE.

D.3 CARD CODES

The RSTS card driver can be configured for one of three different punched card codes. These are: DECØ29 codes, DECØ26 codes and 14Ø1 (EBCDIC) codes. The RSTS-11 DECØ29 and DECØ26 codes are the same as the DOS-11 card codes. The particular set of codes used on the system is determined by the system manager. In all cases, the end-of-file (EOF) card must contain a 12-11-Ø-1 punch or a 12-11-Ø-1-6-7-8-9 punch in column Ø.

CHARACTER	ASCII$_{10}$	DEC029	DEC026	1401
{	123	12 0	12 0	UNUSED
}	125	11 0	11 0	UNUSED
SPACE	32	NONE	NONE	NONE
!	33	12 8 7	12 8 7	11 0
"	34	8 7	0 8 5	0 8 2
#	35	8 3	0 8 6	8 3
$	36	11 8 3	11 8 3	11 8 3
%	37	0 8 4	0 8 7	0 8 4
&	38	12	11 8 7	12
'	39	8 5	8 6	12 8 4
(	40	12 8 5	0 8 4	8 7
)	41	11 8 5	12 8 4	0 8 7
*	42	11 8 4	11 8 4	11 8 4
+	43	12 8 6	12	0 8 5
,	44	0 8 3	0 8 3	0 8 3
-	45	11	11	11
.	46	12 8 3	12 8 3	12 8 3
/	47	0 1	0 1	0 1
0	48	0	0	0
1	49	1	1	1
2	50	2	2	2
3	51	3	3	3
4	52	4	4	4
5	53	5	5	5
6	54	6	6	6
7	55	7	7	7
8	56	8	8	8
9	57	9	9	9
:	58	8 2	11 8 2	8 5
;	59	11 8 6	0 8 2	11 8 6
<	60	12 8 4	12 8 6	12 8 6
=	61	8 6	8 3	11 8 7
>	62	0 8 6	11 8 6	8 6
?	63	0 8 7	12 8 2	12 0

CHARACTER	ASCII$_{10}$	DEC029	DEC026	1401
@	64	8 4	8 4	8 4
A	65	12 1	12 1	12 1
B	66	12 2	12 2	12 2
C	67	12 3	12 3	12 3
D	68	12 4	12 4	12 4
E	69	12 5	12 5	12 5
F	70	12 6	12 6	12 6
G	71	12 7	12 7	12 7
H	72	12 8	12 8	12 8
I	73	12 9	12 9	12 9
J	74	11 1	11 1	11 1
K	75	11 2	11 2	11 2
L	76	11 3	11 3	11 3
M	77	11 4	11 4	11 4
N	78	11 5	11 5	11 5
O	79	11 6	11 6	11 6
P	80	11 7	11 7	11 7
Q	81	11 8	11 8	11 8
R	82	11 9	11 9	11 9
S	83	0 2	0 2	0 2
T	84	0 3	0 3	0 3
U	85	0 4	0 4	0 4
V	86	0 5	0 5	0 5
W	87	0 6	0 6	0 6
X	88	0 7	0 7	0 7
Y	89	0 8	0 8	0 8
Z	90	0 9	0 9	0 9
[	91	12 8 2	11 8 5	12 8 5
\	92	0 8 2	8 7	0 8 6
]	93	11 8 2	12 8 5	11 8 5
↑ or ^	94	11 8 7	8 5	unused
← or _	95	0 8 5	8 2	12 8 7

EOF is 12-11-0-1 punch or a 12-11-0-1-6-7-8-9 punch.

D.4 RADIX-5Ø CHARACTER SET

Character	ASCII Octal Equivalent	Radix-5Ø Equivalent
space	4Ø	Ø
A-Z	1Ø1 - 132	1 - 32
$	44	33
.	56	34
unused		35
Ø-9	6Ø - 71	36 - 47

The maximum Radix-5Ø value is, thus,

$$47*5Ø^2 + 47*5Ø + 47 = 174777$$

The following table provides a convenient means of translating between the ASCII character set and its Radix-5Ø equivalents. For example, given the ASCII string X2B, the Radix-5Ø equivalent is (arithmetic is performed in octal):

```
  X = 113ØØØ
  2 = ØØ24ØØ
  B = ØØØØØ2
X2B = 1154Ø2
```

Radix-5Ø Character/Position Table

Single Char. or First Char.		Second Character		Third Character	
A	ØØ31ØØ	A	ØØØØ5Ø	A	ØØØØØ1
B	ØØ62ØØ	B	ØØØ12Ø	B	ØØØØØ2
C	Ø113ØØ	C	ØØØ17Ø	C	ØØØØØ3
D	Ø144ØØ	D	ØØØ24Ø	D	ØØØØØ4
E	Ø175ØØ	E	ØØØ31Ø	E	ØØØØØ5
F	Ø226ØØ	F	ØØØ36Ø	F	ØØØØØ6
G	Ø257ØØ	G	ØØØ43Ø	G	ØØØØØ7
H	Ø31ØØØ	H	ØØØ5ØØ	H	ØØØØ1Ø
I	Ø341ØØ	I	ØØØ55Ø	I	ØØØØ11
J	Ø372ØØ	J	ØØØ62Ø	J	ØØØØ12
K	Ø423ØØ	K	ØØØ67Ø	K	ØØØØ13
L	Ø454ØØ	L	ØØØ74Ø	L	ØØØØ14
M	Ø5Ø5ØØ	M	ØØ1Ø1Ø	M	ØØØØ15
N	Ø536ØØ	N	ØØ1Ø6Ø	N	ØØØØ16
O	Ø567ØØ	O	ØØ113Ø	O	ØØØØ17
P	Ø62ØØØ	P	ØØ12ØØ	P	ØØØØ2Ø
Q	Ø651ØØ	Q	ØØ125Ø	Q	ØØØØ21
R	Ø7Ø2ØØ	R	ØØ132Ø	R	ØØØØ22
S	Ø733ØØ	S	ØØ137Ø	S	ØØØØ23
T	Ø764ØØ	T	ØØ144Ø	T	ØØØØ24
U	1Ø15ØØ	U	ØØ151Ø	U	ØØØØ25
V	1Ø46ØØ	V	ØØ156Ø	V	ØØØØ26
W	1Ø77ØØ	W	ØØ163Ø	W	ØØØØ27
X	113ØØØ	X	ØØ17ØØ	X	ØØØØ3Ø
Y	1161ØØ	Y	ØØ175Ø	Y	ØØØØ31
Z	1212ØØ	Z	ØØ2Ø2Ø	Z	ØØØØ32
$	1243ØØ	$	ØØ2Ø7Ø	$	ØØØØ33
.	1274ØØ	.	ØØ214Ø	.	ØØØØ34
unused	1325ØØ	unused	ØØ221Ø	unused	ØØØØ35
Ø	1356ØØ	Ø	ØØ226Ø	Ø	ØØØØ36
1	14Ø7ØØ	1	ØØ233Ø	1	ØØØØ37
2	144ØØØ	2	ØØ24ØØ	2	ØØØØ4Ø
3	1471ØØ	3	ØØ245Ø	3	ØØØØ41
4	1522ØØ	4	ØØ252Ø	4	ØØØØ42
5	1553ØØ	5	ØØ257Ø	5	ØØØØ43
6	16Ø4ØØ	6	ØØ264Ø	6	ØØØØ44
7	1635ØØ	7	ØØ271Ø	7	ØØØØ45
8	1666ØØ	8	ØØ276Ø	8	ØØØØ46
9	1717ØØ	9	ØØ3Ø3Ø	9	ØØØØ47

RSTS-11 System
User's Guide
DEC-11-ORSUA-D-D

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

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Documentation
146 Main Street ML5-5/E39
Maynard, Massachusetts 01754

digital equipment corporation

Printed in U.S.A.