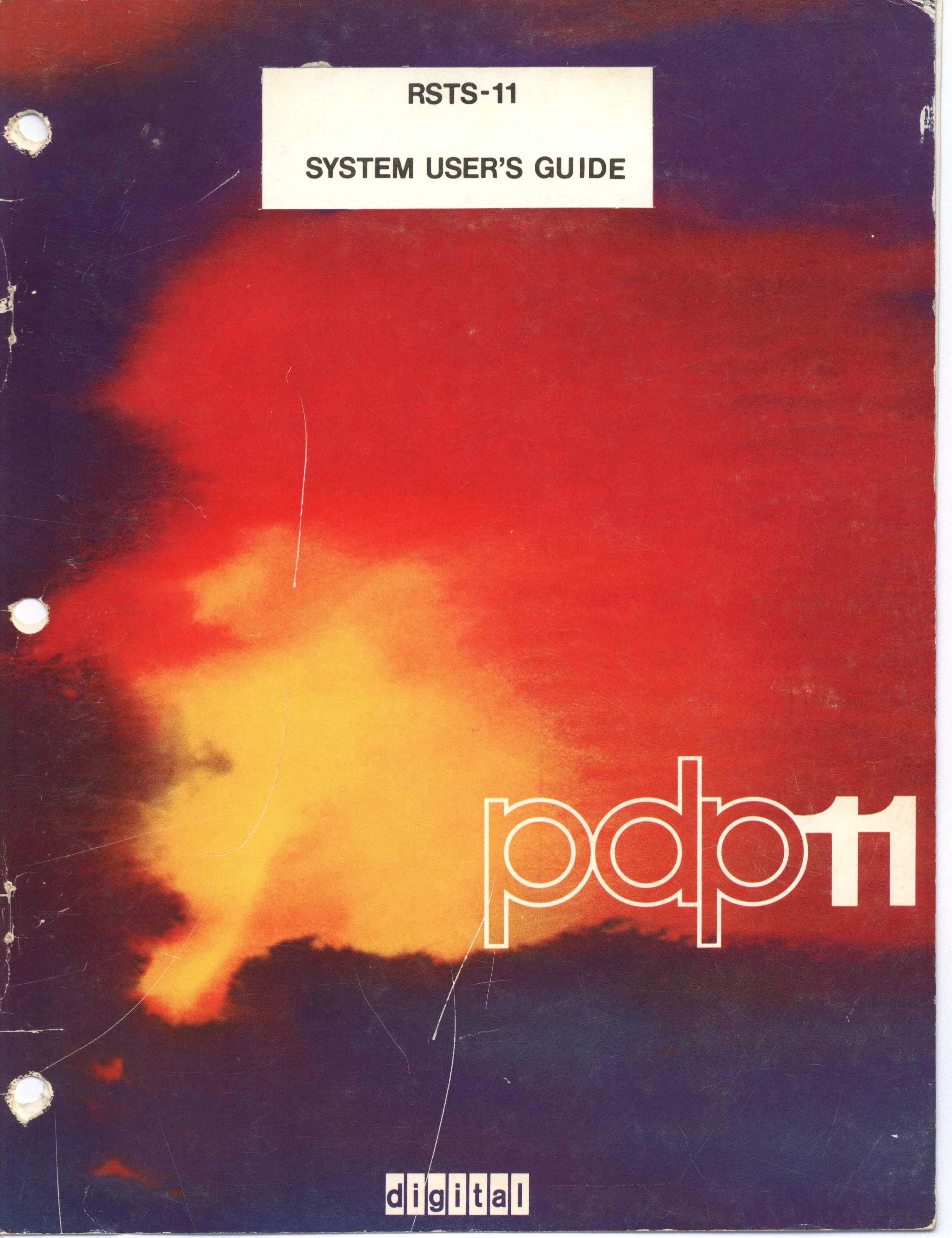
RSTS-11

SYSTEM USER'S GUIDE



pdp11

digital

DEC-11-ORSUA-A-D

RSTS-11

SYSTEM USER'S GUIDE

FOR USE WITH RSTS-11
Version VØØ4A
(PDP-11 RESOURCE TIME-SHARING SYSTEM)

September 1972

<u>SOFTWARE SUPPORT CATEGORY</u>

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS SUPPORTED BY DEC UNDER CATEGORY I AS DEFINED ON PAGE IV OF THIS DOCUMENT.

FOR ADDITIONAL COPIES, ORDER NO. DEC-11-ORSUA-A-D FROM DIGITAL EQUIPMENT CORPORATION, SOFTWARE DISTRIBUTION CENTER, MAYNARD, MASSACHUSETTS 01754

First Printing, September 1972

Your attention is invited to the last two pages of this document. The Reader's Comments page, when completed and returned, is beneficial to both you and DEC. All comments received are considered when documenting subsequent manuals. The How To Obtain Software Information page offers you a means of keeping up to date with DEC's software.

Associated Documents:

BASIC-PLUS Language Manual
DEC-11-ORBPA-A-D

Preface

This manual deals with the interaction between the user and the RSTS-11 terminal, how to enter and edit user programs, and how to give commands to BASIC-PLUS.

The operations of listing, running, saving, or compiling a program differ in kind from the individual statements composing the program. System commands cause these operations to be performed. Other system operations include assigning of peripheral devices for program input and/or output, determining the length of the current program or number of user files available, renaming and replacing of current files, and enabling and disabling of the echo feature on the RSTS-11 terminal.

SOFTWARE SUPPORT CATEGORIES

Digital Equipment Corporation (DEC) makes available four categories of software. These categories reflect the types of support a customer may expect from DEC for a specified software product. DEC reserves the right to change the category of a software product at any time. The four categories are as follows:

CATEGORY I
Software Products Supported at no Charge

This classification includes current versions of monitors, programming languages, and support programs provided by DEC. DEC will provide installation (when applicable), advisory, and remedial support at no charge. These services are limited to original purchasers of DEC computer systems who have the requisite DEC equipment and software products.

At the option of DEC, a software product may be recategorized from Category I to Category II for a particular customer if the software product has been modified by the customer or a third party.

CATEGORY II
Software Products that Receive Support for a Fee

This category includes prior versions of Category I programs and all other programs available from DEC for which support is given. Programming assistance (additional support), as available, will be provided on these DEC programs and non-DEC programs when used in conjunction with these DEC programs and equipment supplied by DEC.

CATEGORY III
Pre-Release Software

DEC may elect to release certain software products to customers in order to facilitate final testing and/or customer familiarization. In this event, DEC will limit the use of such pre-release software to internal, non-competitive applications. Category III software is only supported by DEC where this support is consistent with evaluation of the software product. While DEC will be grateful for the reporting of any criticism and suggestions pertaining to a pre-release, there exists no commitment to respond to these reports.

CATEGORY IV
Non-Supported Software

This category includes all programs for which no support is given

CONTENTS

CHAPTER 1

INTRODUCTION TO RSTS-11

RSTS-11 (Resource Sharing Timesharing System for the PDP-11 computer) is a powerful timesharing system that can support up to 16 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.

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, the user 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-10 to handle 20, 50, 100 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 languages and 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 Teletype.[1] 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.

[1] Teletype is a registered trademark of the Teletype Corporation.

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 2.4.3 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 Teletype 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]This knob is found on ASR-33 and ASR-35 Teletypes. Alternate types of user terminals may have a different knob or switch designed to put the device on-line.

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 V4A-11 SYSTEM #213  JOB 1  KB2  15-AUG-72  12:28 AM
#100,101
PASSWORD:

WELCOME TO RSTS-11 !

DEMO PROGRAMS ON THIS SYSTEM ARE:
        $DEMO1 -- ORDER ENTRY AND INVENTORY CONTROL.
        $DEMO2 -- PAYROLL AND CHECK-PRINTING.




NEW OR OLD--
```

Once successfully logged onto the system, the user can type NEW to create a new program, OLD to retrieve a program previously saved, or any other RSTS-11 command.

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.

```
HELLO 1ØØ,1Ø2

RSTS V4A-11  SYSTEM #213   JOB 1   KB2   15-AUG-72   12:29 AM
PASSWORD:


WELCOME TO RSTS-11 !

DEMO PROGRAMS ON THIS SYSTEM ARE:
     $DEMO1 -- ORDER ENTRY AND INVENTORY CONTROL.
     $DEMO2 -- PAYROLL AND CHECK-PRINTING.




NEW OR OLD--
```

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

RSTS V4A-11  SYSTEM #213   JOB 1   KB2   15-AUG-72   12:32 AM
#1ØØ/1ØØ
PASSWORD:


NEW OR OLD--
```

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 NEW OR OLD--
I	Individual. The user wishes to individually examine each of his files. The system prints a listing 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.)
RETURN Key	Causes the system to print a message explaining that typing a ? character explains acceptable replies.
?	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.

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

```
BYE
CONFIRM: Y
SAVED ALL DISK FILES; 387 BLOCKS IN USE, 113 FREE
JOB 1 USER 1ØØ,1Ø1 LOGGED OFF KB2 AT 15-AUG-72 12:28 AM
SYSTEM RSTS V4A-11 SYSTEM #213
RUN TIME WAS .7 SECONDS
ELAPSED TIME WAS 45 SECONDS
GOOD MORNING
```

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
 OTHER   LOG ME OUT
CONFIRM: N


NEW OR OLD--
```

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
CONFIRM: Y
DISK QUOTA OF 5ØØ EXCEEDED BY 64 BLOCKS
SOME FILE(S) MUST BE DELETED BEFORE LOGGING OUT
CONFIRM: I
NONAME.BAS          1        6Ø       14-AUG-72   ? K
TRASH .             553      6Ø       14-AUG-72   ? K
FTI   .             1Ø       6Ø       15-AUG-72   ?
CONFIRM: Y
SAVED ALL DISK FILES; 1Ø BLOCKS IN USE, 49Ø FREE
JOB 1 USER 1ØØ,1Ø2 LOGGED OFF KB2 AT 15-AUG-72 12:31 AM
SYSTEM RSTS V4A-11 SYSTEM #213
RUN TIME WAS 1.7 SECONDS
ELAPSED TIME WAS 2 MINUTES, 45 SECONDS
GOOD MORNING
```

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 Teletype 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, instructions refer to the ASR-33 and ASR-35 Teletype terminal. 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.

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
1Ø LET A ,B  = 25
?2Ø 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 1ØØ
```

causes line number 1ØØ to be deleted.

```
DELETE 1ØØ - 2ØØ
```

causes all program lines between and including line numbers 1ØØ and 2ØØ to be deleted. If 1ØØ and/or 2ØØ do not exist in the program, any lines within the range from 1ØØ to 2ØØ 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.

2.2.6 <u>RENAME Command</u>

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 debuggin , ¬hase. 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.

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 (e.g. the user has modified the memory through immediate mode statements after stopping), the

```
CAN'T CONTINUE
```

message is printed.

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.

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).

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 or a RUN FILEV3.BAS command. 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 if 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, that filename is used as the current program name when the program is read into memory. However, if no filename is specified, the file in memory will have no name and no SAVE command, or any other command referencing a filename, can be issued without assigning a name to that file (see the RENAME command, Section 2.2.6). The same results occur when issuing the command:

```
RUN CR:
```

NOTE

The card reader can detect the absence of cards in the card reader; however, the paper tape reader is unable to distinguish between the absence of tape in the reader and the reader being off-line. In both cases an:

```
END OF FILE ON DEVICE
```

error is printed, following which a file with no name and containing no statements exists in the user's memory.

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. An extension, 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 RSTS-11 Run Time System into a form executable on the PDP-11 computer.) 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

---

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

stored with a default protection of <6Ø> . To obtain any other protection code, that code must be explicitly specified. For example:

```
COMPILE <4Ø>
```

creates a file on the system disk having the current filename, a .BAC extension, and a protection code of <4Ø> . At any time protection codes can be altered with the NAME-AS command or the PIP system program.

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:

```
FILE EXISTS-USE REPLACE
```

This message is designed to protect the user against inadvertently destroying an existing file.

Where 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 any 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:

```
SAVE 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 (.BAS files), since compiled programs (.BAC files) can be run but not changed. Any program called with the OLD command can be edited by the user at the terminal.

2.4.3 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 and can be used on a file only by someone logged into the system under the account number which owns the 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.

2.4.4 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 synonomous 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.)

2.4.5 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.

2.4.6 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
```

A minimum 2K of memory (1K = 1ø24 words) is reserved for each user, with a maximum of 8K per user in 28K systems and 6K per user in 24K systems. The size of the current program is reported to the nearest 1K.

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
TTYTST.BAS        4        6ø     26-JUL-72     26-JUL-72     ø2:12 PM
JUNKY1.TST        1        6ø     26-JUL-72     26-JUL-72     ø4:ø6 PM
TEMPø2 .TMP       1        6ø     31-JUL-72     31-JUL-72     ø2:54 PM
PROG   .BAS       1        6ø     31-JUL-72     31-JUL-72     ø2:54 PM
PROG1  .BAS       1        6ø     31-JUL-72     31-JUL-72     ø2:59 PM
PAT    .BAS       1        6ø     31-JUL-72     31-JUL-72     ø3:ø1 PM
 ↑       ↑        ↑         ↑          ↑             ↑            ↑
name  extension  size in 256-  protection  last access date  creation date  creation time
                 word blocks   code
```

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 (generally a BASIC-PLUS program, although a data file could also be input this way). 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 the RETURN and RUBOUT key characters are ignored.

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 ignored 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
LP:	line printer
DTØ: to DT7:	DECtape units Ø to 7
KB:	current user terminal
KBn:	Teletype n in the system

NOTE

the user can reference DTn: and MTn: where n is between Ø and the maximum number of such units on the system. For example, normally only DTØ: and DT1: can be referenced on a RSTS-11 system.

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.

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 <u>not</u> 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 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 LEW A↑U
20 LET A=
23.4↑U
LISTNH
SYNTAX ERROR AT LINE 20

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, the LINE FEED key was used to continue line 2Ø onto a second line. A physical line has been terminated. 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:

```
2Ø LET A =
```

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

```
2Ø LET A =
23.4
```

which would be accepted as equivalent to:

```
2Ø LET A = 23.4
```

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 output is occuring 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.)

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).

```
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

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.
GRIPE	allows the user to log complaints or comments about the system.
QUOLST	allows the user to determine his current disk quota and the amount of file space in each system device.
MONEY	allows the user to determine the amount of time charged to his account number.

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.)

4.1 LOGIN PROGRAM

The LOGIN program is started when the command HELLO is typed at a user terminal or when any line is entered from a logged-out terminal. LOGIN connects a user terminal to RSTS-11, attaches a user to another job already running within the system, or permits the user to run designated system programs from a logged-out terminal. LOGIN can be called when the user first logs into the system or after the user is already logged into the system.

The following are acceptable inputs from a logged-out terminal:

Input	Meaning	Manual Section
HELLO	User wants to log into the system.	4.1
HELP	User wants a printed description of how to use the system.	
I	Dataset has been answered and wants to log into the system. If the system has an automatic answering facility for remote terminals, the I character is inserted into the user input buffer by the system.	
other	Undefined. LOGIN prints the message: PLEASE SAY HELLO.	
SYS	User wants a system status report.	4.3
SET xxxx	User wants to set characteristics for that terminal.	4.5

If the user input is SET xxxx or SYS, RSTS-11 attempts to run the system program requested. Only those programs listed above or others specified by the system manager can run while logged out of the system (see section 2.1.4).

In response to the HELLO command or the automatically inserted I character, where the issuing terminal is not logged into the system, the LOGIN program is started and performs the following:

a. LOGIN prints the following information at the user terminal: the job number, keyboard number, system version, and current system date and time.

b. LOGIN then requests the user account number by printing a # character. The user types his project and programmer numbers in that order separated by a comma and terminated with the RETURN key. (Separating the numbers with a slash inhibits the printing of any opening system messages.) The user can specify his project-programmer numbers on the same line as the HELLO command; for example:

```
HELLO   1ØØ,1ØØ
```

(In this case, the system does not print # and only prompts the user to type his password, as shown below.)

c. LOGIN then requests a password by printing:

```
PASSWORD:
```

The password typed by the user does not echo at the terminal.

d. If the account number and password are a valid entry, the user is logged into the system. If the codes typed by the user are not an acceptable entry, the message:

```
INVALID ENTRY - TRY AGAIN
#
```

is printed and the user can try entering another account combination.

e. If there are any detached jobs[1] with the current account number, LOGIN then informs the user and permits him to attach to one of those jobs (thus killing his current job number and removing it from the system). See the description below of attaching to another job.

f. If the user does not attach to another job, LOGIN prints the message of the day, if any.

g. Finally, the system prints

```
NEW OR OLD - -
```

to indicate that it is ready to accept command input.

The complete sequence is shown below:

```
HELLO

RSTS V4A-11 SYSTEM #213   JOB 3   KBØ   Ø5-SEP-72   Ø4:25 AM
#1ØØ, 1ØØ
PASSWORD:

WELCOME TO RSTS-11  !

NEW OR OLD--
```

---

[1]A job becomes detached either because of a dataphone disconnect affecting a remote user or due to deliberate action by deliberate use of a certain system function within that job program.

If the HELLO command is input from a terminal already logged into the system, LOGIN performs the following:

a. LOGIN prints two lines identifying the system version, job number, keyboard, date, time and account number.

b. If there are any detached jobs with the current account number, LOGIN permits the user to attach to one of those jobs (which kills his current job number and removes it from the system). The job number to be typed by the user is the job number as printed by the system following the HELLO command. For example:

```
HELLO
RSTS-11  V4A-Ø5 SYSTEM #213  JOB 3  KB4  1Ø-SEP-72  1Ø:ØØ AM
YOU ARE LOGGED IN UNDER 1ØØ,1ØØ
JOB(S)  2  ARE DETACHED UNDER THIS ACCOUNT
TYPE JOB NUMBER TO ATTACH TO?  2
ATTACHING TO JOB  2
```

The job number specified must be the original job number assigned when the currently detached job was logged into the system.

c. To avoid attaching to another job, type the RETURN key instead of specifying a job number.

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 give any of the following responses:

Table 4-1

LOGOUT CONFIRM: Responses

CONFIRM: Response	Meaning
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 NEW or OLD--
?	Causes the system to print an explanation of the acceptable responses to CONFIRM:
RETURN key	Causes the system 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	Causes the user to be logged out of the system as long as his disk storage space is within the acceptable limits.

In individual file 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
CONFIRM:  YES
DISK QUOTA OF 5ØØ  EXCEEDED BY 1Ø  BLOCKS
SOME FILE(S) MUST BE DELETED BEFORE LOGGING OUT
CONFIRM:  I
FOOBAR.BAS        45Ø      6Ø       1Ø-JAN-72     ?
FOO    .TMP       6Ø       6Ø       1Ø-JAN-72     ?  K
CONFIRM:  YES
SAVED ALL DISK FILES:  45Ø  BLOCKS IN USE,  5Ø  FREE
JOB 1  USER  1ØØ,1ØØ  LOGGED OFF KB4 AT 1Ø-JAN-72  1Ø:ØØ AM
SYSTEM RSTS-11 V4A-Ø5  SYSTEM #213
RUN TIME WAS 1 MINUTE.  2Ø.1  SECONDS
ELAPSED TIME WAS  2Ø  MINUTES.  15  SECONDS
GOOD MORNING
```

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

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.
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
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[1]
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? 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.

The following examples are performed on a terminal <u>not</u> 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 SBDF	equivalent to not specifying any options; a complete system status report is output to the terminal.
SYS 5	causes output of a status report for job 5 to the terminal.

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

A complete system status report is shown below:

```
RUN$SYSTAT
OUTPUT STATUS TO?

RSTS V4A-11 SYSTEM #213 STATUS ON Ø5-SEP-72 AT Ø4:42 AM UP: 1:34:42

JOB        WHO       WHERE    WHAT      SIZE    STATE     RUN-TIME
 1       1ØØ,1Ø1     KB6     LOGOUT      2K     RN  SW        6.7
 2       1ØØ,1ØØ     KB1Ø    NONAME      2K     KB            Ø.9
 3       1ØØ,1ØØ     KBØ     SYSTAT      5K     RN            Ø.9
 4         1, 1Ø     KB9     MONEY       5K     FP            Ø.3

BUSY DEVICES:
DEVICE   JOB    WHY
LP        1     AS

DISK STRUCTURE:
DISK     OPEN     FREE    CLUSTER  ERRORS    COMMENTS
DKØ       Ø       686       1        Ø        PUBLIC
DK1       2       139       2        Ø        PRIVATE

SMALL   LARGE    ERRORS    HUNG TTY'S
 43       1        Ø           Ø

READY
```

Run time is given in

hours: minutes: seconds. tenths of seconds

For example:

1:23.4

indicates one minute, 23.4 seconds of run time.

The job status information includes a list of all current active jobs by job number. Associated with each job number are the account number, the keyboard number, the current program name and size, the job state and elapsed run time associated with that job.

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, catastrophic error count, 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 following an I/O operation
HB	job is detached (waiting to be attached)
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

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; the only difference being that one contains Record I/O copy options and is present only on systems having the Record I/O feature. Non-Record I/O PIP can perform only formatted ASCII file transfers. (RSTS-11 PIP commands, wherever possible, have been made compatible with DOS-11 PIP commands).

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

```
RUN$ PIP
```

PIP responds by identifying itself and printing an asterisk to indicate that it is able to accept input commands:

```
PIP   RSTS  V4A-11  SYSTEM #213
*
```

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, as does CTRL/C.

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 8Ø 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.

Table 4-4

Output File Specification Elements

Element	Description	Default
dev:	device specification	system disk (DF:)
[proj,prog]	account specification, project-programmer number	current user account number
name	filename specification[1]	none
.ext	filename extension[1]	none
< prot >	protection code	< 6Ø >, 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:)
[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 files 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:

```
RUN$ PIP
PIP RSTS V4A-11 SYSTEM #213
*DT1:FILET.BAS<FILE1,FILE2,PR:,DTØ: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. The general format for such a command is as follows:

```
*new file specification = old file specification
```

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.[1]
/BL:n	Block mode transfer is performed using a block which is n bytes long.[1]
/CO	Contiguous mode transfer is performed with null fill characters inserted into any partial buffer remaining.[2]
/CO:T	Contiguous mode transfer is performed with any partial buffer remaining being truncated.[2]
/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).

[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 specification. The current protection code need not be indicated. For example:

```
*FILE.EXT<4Ø> =ABC.DAT
```

In this example, the filename, extension, and protection code are changed. The file is stored on the system disk, the default device.

```
*DT1:MAT.BAC=DT1:ORIG.BAC
```

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>
```

or

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

where the /RE (rename) option is assumed if not specified. This command string format indicates that the file specification is to be updated. For example:

```
* = MAG.BAS <48>
```

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:

```
RUN$ PIP
PIP RSTS V4A-11 SYSTEM #213
*/ZE
```

This command removes all files stored on the system disk for the current account number.

```
*DT1: /ZE
```

removes all files on the tape reel mounted on DECtape unit 1. (No file protection is available for files on DECtape reels; hence, any user can zero any DECtape reel.)

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 V4A-11 SYSTEM #213
*, DTØ:, DT1:/DI
```

will cause 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.

Sample directory listing specifications are shown below:

```
*/BR
*DTØ : /DI
* *.BAC/DI:S
* $/DI
```

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 Table 4-9. 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 4-9

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-33 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-33 Teletypes).	Set
LC	Enable sending 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
ESC	Treat only ASCII Ø33 (octal) code as ESCAPE (ESCAPE or ALT MODE key).	
NO ESC	Treat ASCII Ø33 , 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 DC11baud rate to DC11speed n where $Ø \leq n \leq 3$ (the actual baud rate corresponding to n is dependent upon the particular DC11 referenced.	SPEED Ø

Table 4-9

TTYSET Commands (cont'd.)

Command	Function	Standard Condition
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 Ø.	Set
FILL LA3Ø	Set special LA3Ø serial (DECwriter) fill factor.	
KSR33 ASR33 KSR35 ASR35 VTØ5 VTØ6 LA3Ø LA3ØS	Treat terminal as though it were the indicated device. These commands automatically select, from the preceeding commands, those features applicable to the respective terminal.	Set

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 4-10

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).

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:

```
RUN$ MONEY

 ACCT    PASSWORD  CPU-TIME  KCT'S CONNECT DEVICE   ACCESS  DISK QUOTA
100,100               57.5   2570  24:33        0    8   7   300   500  2

READY
```

CHAPTER 5

RSTS-11 PERIPHERAL DEVICES

RSTS-11 has several peripherals which are available to the user. These devices include:

user terminals (ASR-33, ASR-35, DECwriter, VTØ5 display)
high-speed paper tape reader/punch
card reader
line printer
DECtape
magtape

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. Chapter 12 in the BASIC-PLUS Language Manual describes device usage for the transfer of information between peripheral equipment and the BASIC-PLUS program.

5.1 ASR-33 Teletype

The ASR-33 Teletype is an inexpensive, commonly employed user terminal. Major features are noted in Figure 5-1.

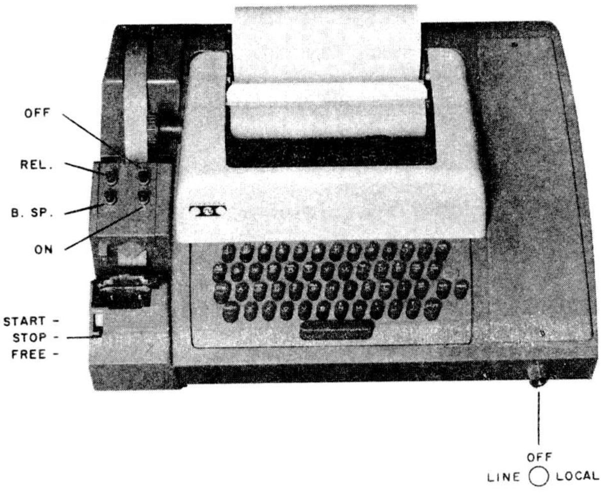


Figure 5-1 ASR-33 Teletype Console

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]

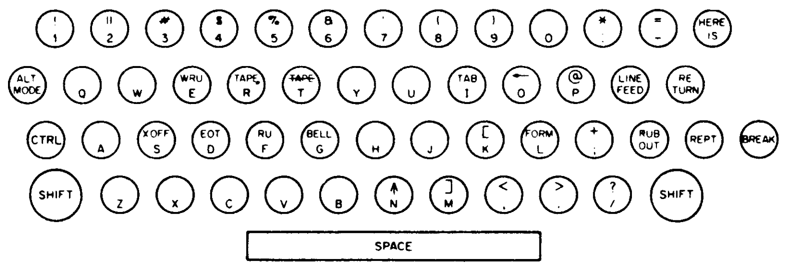


Figure 5-2 Teletype Keyboard

[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.

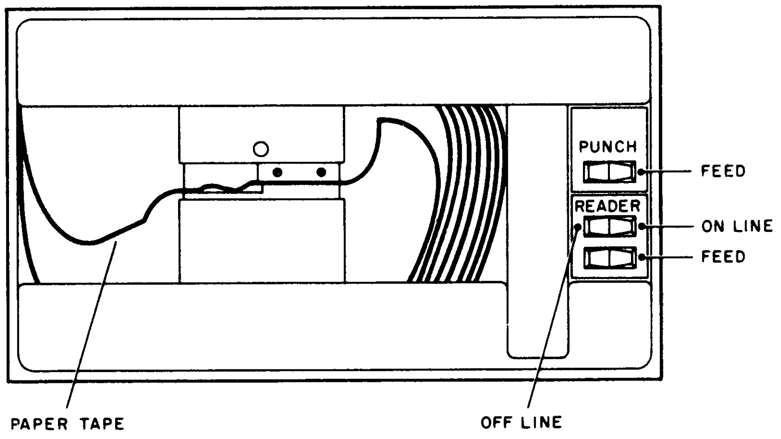


Figure 5-3 High-Speed Paper Tape Reader/Punch

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 8∅ column data cards. Cards can be read under program control at rates of up to 2∅∅ or 3∅∅ 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.

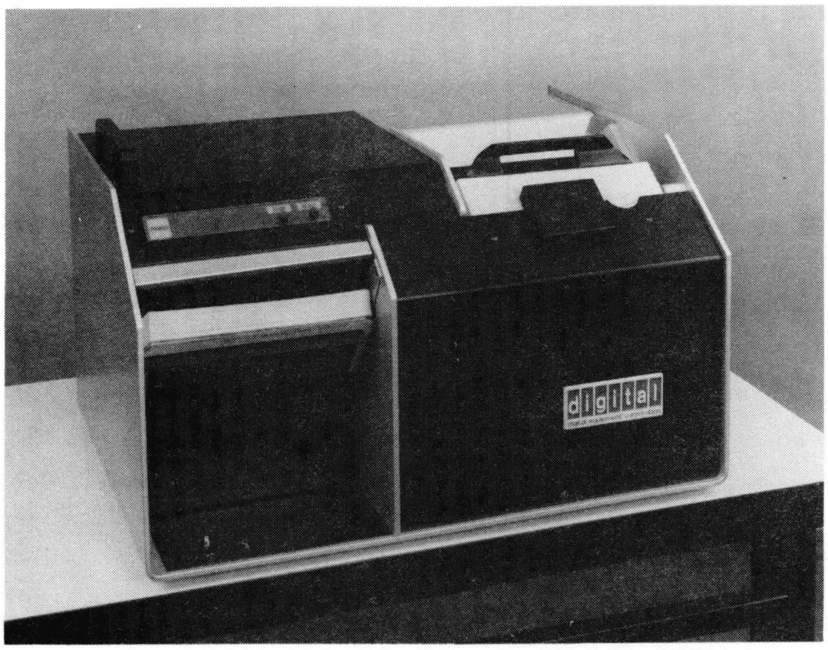


Figure 5-4

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.

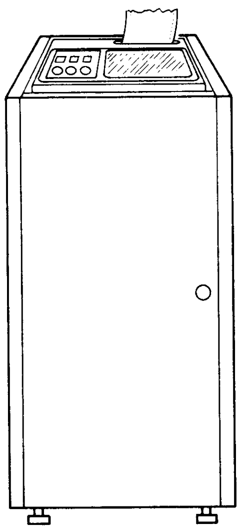


Figure 5-5
LP11 Line Printer System (8Ø-column model)

5.4.1 Line Printer Character Set

The 64 character set consists of the 26 upper case letters (A - Z), ten numerals (Ø - 9), 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.

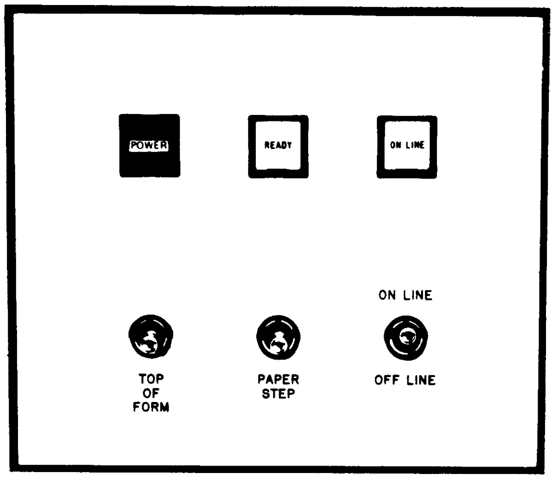


Figure 5-6 Line Printer Control Panel

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.)

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 8Ø 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 1Ø 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.

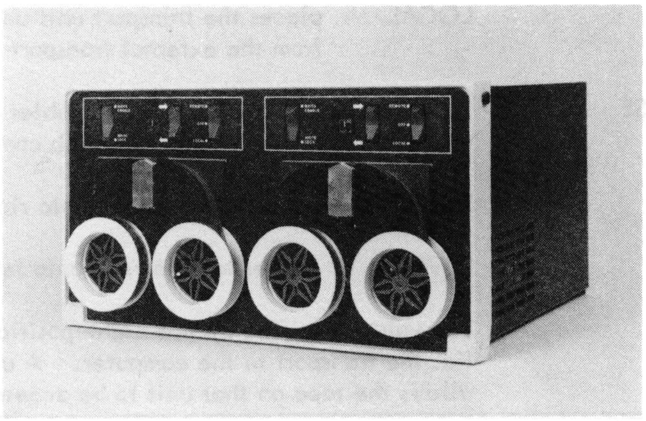


Figure 5-7 TU56 DECtape Transport

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.)

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, <u>and</u> 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 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 the TU10 Controller reads and writes information in an industry-compatible format.

The basic DECmagtape system consists of three components:

a. TU10 Tape Transport, pictured in Figure 5-8, can read or write information on magnetic tape in nine channels (on RSTS-11, with a choice of 7 or 9 channels in other applications).

b. TM11 Controller is an interface between the tape transport units and the RSTS-11 system (unibus). One Controller serves up to eight transport units. The Controller is transparent to the RSTS-11 user.

c. Magnetic Tape, the recording medium used for data storage, consists of reel-mounted magnetic tape formatted in 9-channel industry-compatible format. Tape includes end-of-tape (EOT) and beginning-of-tape (BOT) reflective (silver) markers, record gaps, lateral and longitudinal parity characters, and file marks, with a cyclic redundancy check character.

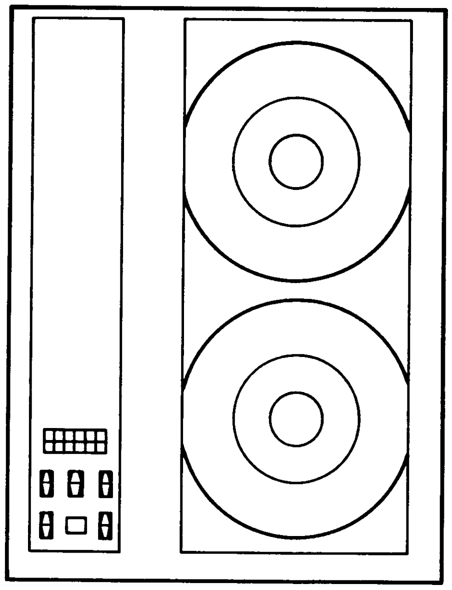


Figure 5-8 TU10 DEC Magnetic Tape System

Transfer rate is up to 36,ØØØ characters per second. 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 TU10 magtape transport control panel is shown in Figure 5-9. This panel is located at the lower left of the TU10 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.

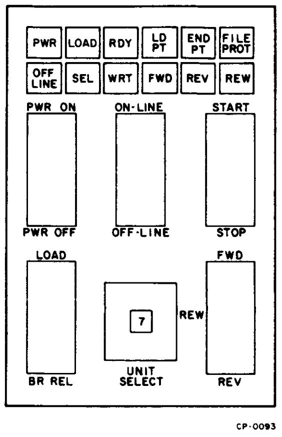


Figure 5-9 TU1Ø Control Panel

Table 5-4

Magtape Transport Controls

Switch	Function
PWR ON/ PWR OFF	This two-position switch applies power to the transport unit.
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. Setting to OFF-LINE causes tape to wind to BOT marker.
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 EOT marker is sensed. REW rewinds tape onto the feed reel until BOT marker is sensed; depressing the switch again causes the tape to completely rewind. REV rewinds tape until BOT marker is sensed.

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.)

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 by depressing the PWR ON switch. Ensure that the LOAD/BR REL switch is in the center position. Transport is ON-LINE.

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.

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 5-8. 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.

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 1Ø 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.

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.

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	== undefined for strings
	<>	not equal to	
	==	approximately equal to	
Logical	NOT	logical negation	relational expressions composed of string or numeric elements with relational operators
	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 of 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	

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 in radians.
	Y=COS(X)	returns the cosine of X 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=LOG1Ø(X)	returns the common logarithm of X, $\log_{1Ø} X$.
	Y=PI	has a constant value of 3.14159.
	Y=RND	returns a random number between Ø and 1.
	Y=RND(X)	returns a random number between Ø 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 in radians.
	Y=SQR(X)	returns the square root of X.
	Y=TAN(X)	returns the tangent of X in radians.
Print	Y%=POS(X%)	returns the current position of the print head for I/O channel X, Ø is the user's Teletype. (This value is imaginary for disk files.)
	Y$=TAB(X%)	moves print head to position X in the current print record, or is disregarded if the current position is beyond X. The first position is counted as Ø.)
String	Y%=ASCII(A$)	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%$(I%)	maps integer into 2-character string, see Section 11.5.
	Y$=CVTF$(X)	maps floating-point number into 4- or 8-character string, see Section 11.5.
	Y%=CVT$%(A$)	maps first 2 characters of string A$ into an integer, see Section 11.5.
	Y=CVT$F(A$)	maps first 4 or 8 characters of string A$ into a floating-point number. See Section 11.5.

Type	Function	Explanation
String, cont'd.	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$, see Section 11.7.
System	Y$=DATE$(Ø%)	returns the current date in the following format: Ø2-Mar-71
	Y$=DATE$(N%)	returns a character string correspond- to a calendar date as follows: N=(day of year)+[(number of years since 1970)*1000] DATE$(1) = "Ø1-Jan-7Ø" DATE$(24Ø) = "Ø5-May-7Ø"
	Y$=TIME$(Ø%)	returns the current time of day as a character string as follows: TIME$(Ø) = "Ø5:3Ø PM"

Type	Function	Explanation
	Y$=TIME$(N%)	returns a string corresponding to the time at N minutes before midnight, for example: TIME$(1) = "11:59 PM" TIME$(144∅) = "12:∅∅ AM" TIME$(721) = "11:59 AM"
	Y=TIME(∅%)	returns the clock time in seconds since midnight, as a floating point number.
	Y=TIME(1%)	returns the central processor time used by the current job in tenths of seconds.
	Y=TIME(2%)	returns the connect time (during which the user is logged into the system) for the current job in minutes.
	Y%=ERR	returns value associated with the last encountered error if an ON ERROR GOTO statement appears in the program. See Section 8.4.
	Y%=ERL	returns the line number at which the last error occurred if an ON ERROR GOTO statement appears in the program. See Section 8.4.3.
	Y%=SWAP%(N%)	causes a byte swap operation on the two bytes in the integer variable N%.
	Y$=RAD$(N%)	converts an integer value to a 3-character string and is used to convert from Radix-5∅ 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 elements entered in the last row.
	Y%=NUM2	following input of a matrix, NUM2 contains the number of rows input.
Magtape	Y%=MAGTAPE(I1%,I2%,I3%)	provides program control over magtape operations by means of several function specifications. See Section 12.3.7.
	Y%=RECOUNT	returns the number of characters read following every input operation. Used primarily with non-file structured devices. See Section 11.3.1.

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.

```
Statement Formats and Examples                         BASIC-PLUS Language
                                                       Manual Section

REM
   {line number}  REM <message>                              3.1
   {line number}{<statement>}!<message>
           1Ø  REM  THIS IS A COMMENT
           15  PRINT   !PERFORM A CR/LF

LET
   {line number}{LET}<var>{,<var>,<var>...} = <exp>          3.2
           55  LET A=4Ø: B=22
           6Ø  B,C,A=4.2      !MULTIPLE ASSIGNMENT

DIM
    line number   DIM<var(dimension(s))>                     3.6.2
           1Ø     DIM A(2Ø), B$(5,1Ø), C%(45)                  7.1

    line number   DIM #<constant>,<var(dimension(s))>=<constant>9.6.1
           75     DIM #4, A$(1ØØ)=32,B(5Ø,5Ø)                9.6.2


RANDOMIZE                                                    3.7.2
    line number  RANDOM{IZE}
           55    RANDOMIZE
           7Ø    RANDOM
IF-THEN, IF-GOTO          ┌THEN<statement>    ┐
    line number  IF <cond>│THEN<line number> │             3.5
                          └GOTO<line number> ┘
           55    IF A>B OR B>C THEN PRINT "NO"
           6Ø    IF FNA(R)= B THEN 25Ø
           95    IF L<X↑2 AND L<>Ø GOTO 345

IF-THEN-ELSE                                                  8.5
                          ┌THEN<statement>    ┐
    line number  IF <cond>│THEN<line number> │┌ ELSE<statement>    ┐
                          └GOTO<line number> ┘└ ELSE<line number> ┘
           3Ø    IF B=A THEN PRINT "EQUAL" ELSE PRINT "NOT EQUAL"
           5Ø    IF A>N THEN 2ØØ ELSE PRINT A
           75    IF B==R THEN STOP ELSE 8Ø

FOR
    line number  FOR <var>= <exp>TO <exp> {STEP<exp>}        3.6.1

           2Ø    FOR I=2 TO 4Ø STEP 2
           55    FOR N=A TO A+R

FOR-WHILE, FOR-UNTIL                      ┌     ┐             8.6
    line number  FOR <var> = <exp> {STEP<exp>}│WHILE│ <cond>
                                          │UNTIL│
                                          └     ┘
           84    FOR I = 1 STEP 3 WHILE I<X
           74    FOR N = 2 STEP 4 UNTIL N>A OR N=B
           Ø5    FOR B= 1 UNTIL B>1Ø

NEXT                                                         3.6.1
    line number  NEXT <var>
           25    NEXT I
           6Ø    NEXT N
```

```
                                                      BASIC-PLUS Language
Statement Formats and Examples                        Manual Section


DEF, single line                                                3.7.3
        line number  DEF FN<var>(arg) =<exp(arg)>                  5.5.1
                 2Ø  DEF FNA(X,Y,Z)=SQR(X↑2+Y↑2+Z↑2)                 6.4

DEF, multiple line
        line number  DEF FN<var>(arg)                                8.1
                 <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
        line number  ON <exp> GOTO <list of line numbers>          8.2
                 75  ON X GOTO 95, 15Ø, 45, 2ØØ

GOSUB
        line number  GOSUB <line number>                           3.8.1
                 9Ø  GOSUB 2ØØ


ON-GOSUB
        line number  ON <exp> GOSUB <list of line numbers>         8.3
                 85  ON FNA(M) GOSUB 2ØØ, 25Ø, 4ØØ, 375

RETURN
        line number  RETURN                                        3.8.2
                375  RETURN

CHANGE                                                            5.2

      {line number} CHANGE ⎡<array name>⎤  TO ⎡<string var>⎤
                           ⎣<string var>⎦     ⎣<array name>⎦
                 25  CHANGE A$ TO X
                 7Ø  CHANGE M TO R$
                 75  CHANGE B TO B$

OPEN                                                            9.2
      {line number} OPEN<string>{FOR⎡INPUT ⎤}AS FILE <exp>       9.2.1
                                    ⎣OUTPUT⎦                      9.2.2
            {,RECORDSIZE<exp>}{,CLUSTERSIZE <exp>}{,MODE <exp>}
                 1Ø  OPEN "PP:" FOR OUTPUT AS FILE B1             9.6.3
                 2Ø  OPEN "FOO" AS FILE 3
                 3Ø  OPEN "DT4:DATA.TR" FOR INPUT AS FILE 1Ø
CLOSE                                                            9.5
      {line number} CLOSE <list of exp>                          11.2
                1ØØ  CLOSE 2
                255  CLOSE 1Ø, 4, N1
                                                                3.3.1
READ                                                              5.3
        line number  READ <list of variables>                      6.3
                 25  READ A, B$, C%, F1, R2, B(25)                 10.1
```

```
Statement Formats and Examples                                  BASIC-PLUS Language
                                                                Manual Section

DATA                                                                 3.3.1
        line number  DATA <list of values>                             5.3
                3ØØ  DATA 4.3, "STRING",85,49,75.Ø4,1Ø                 6.3

RESTORE                                                              3.3.1
        line number  RESTORE                                          10.2
                125  RESTORE

PRINT                                                                3.3.2
                                                                       5.4
      {line number}  PRINT{{#<exp>,}<list>}                            6.3
                 25  PRINT  !GENERATES  CR/LF                        9.2.3
                 75  PRINT "BEGINNING OF OUTPUT";I,A*I                 9.3
                 45  PRINT #4,"OUTPUT TO DEVICE"FNM(A)↑2;B;A          10.4

PRINT USING
      {line number}  PRINT {#<exp>,}USING <string>, <list>            10.4
                 54  PRINT USING "##.##",A
                 55  PRINT #3, USING"\\###.##   \\##↑↑↑↑","A=",A,"B=",B
                 56  PRINT #7, USING B$,A,B,C

INPUT                                                                3.3.3
      {line number}  INPUT {#<exp>,}<list>                             5.3
                 25  INPUT "TYPE YOUR NAME ",A$                        6.3
                 55  INPUT #8, A, N, B$                              9.2.3
                                                                     9.2.5
                                                                       9.4


INPUT LINE                                                             5.3
      {line number}  INPUT LINE {#<exp>,} <string>                    10.3
                 4Ø  INPUT LINE  R$
                 75  INPUT LINE  #1, E$

NAME-AS                                                                9.7
      {line number}  NAME <string> AS <string>
                455  NAME "NONAME" AS "FILE1<48>"
                27Ø  NAME "DT4:MATRIX" AS "MATA1<48>"

KILL                                                                   9.8
      {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
        line number   CHAIN <string> {<line number>}                   9.9
                375   CHAIN "PROG2.BAC"
                5ØØ   CHAIN "PROG3.BAC" 75
```

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

```
                                        ⎧ZER⎫
    {line number}MAT <matrix name>=     ⎨CON⎬ {dimension(s)}
                                        ⎩IDN⎭
                 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
```

```
Statement Formats and Examples
                                                    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>      11.4.2
           9Ø  LSET B$="XYZ"

   <line number> RSET<string var>{,<string var>}=<string>      11.4.2
          25Ø  RSET C$="6789Ø"

   <line number> FIELD#<expr>,<expr>AS<string var>{,<expr>AS<string var>}
           75  FIELD#2%,1Ø% AS A$, 2Ø% AS B$                   11.4.1

   <line number> GET#<expr>{,RECORD<expr>}                       11.3
          1ØØ  GET#1%,RECORD 99%

   <line number> PUT#<expr>{,RECORD<expr>}{,COUNT<expr>}         11.3
          5ØØ  PUT#1%,COUNT 8Ø%

   <line number> UNLOCK#<expr>                                   12.2
          7ØØ  UNLOCK #3%
```

APPENDIX B

BASIC-PLUS COMMAND SUMMARY

Command	Explanation	Section in RSTS-11 System User's Guide
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.	2.6.3
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
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.	2.6.4
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
HELLO	Indicates to RSTS that a user wishes to log onto the system. Allows the user to input project-programmer number and password.	2.1.2
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

Command	Explanation	Section in RSTS-11 System User's Guide
LENGTH	Returns the length of the user's current program in core, in 1K increments.	2.5.1
LIST	Allows the user to obtain a 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
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
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.	2.4.6
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	Same as RUN, but does not print header containing the program name and current date.	2.3.1

Command	Explanation	Section in RSTS-11 System User's Guide
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
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.	2.4.5

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/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

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:

```
10  TALK
ILLEGAL VERB AT LINE 10
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 or an attempt was made to perform a directory oriented access to a non-directory device.

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.
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.
30-41	not assigned	
42	VIRTUAL BUFFER TOO LARGE	Virtual core buffers must be 512 bytes long.
43	VIRTUAL ARRAY NOT ON DISK	A non-disk device is open on the channel upon which the virtual array is referenced.

ERR	Message Printed	Meaning
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	Floating point overflow or underflow. If no transfer is made to an error handling routine, a Ø 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	not assigned	
51	INTEGER ERROR	Attempt to use a number as an integer when that number is outside the allowable integer range. If no transfer is made to an error handling routine, a Ø is returned as the integer value. (C)
52	ILLEGAL NUMBER	Improperly formed input or value. For example, "1..2" is an improperly formed number.
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-USE 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.
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
Ø through 9
Space
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).
' "	Used to delimit string constants, i.e., text strings (Section 5.1).
!	Begins comment part of a line (Section 3.1).
:	Separates multiple statements on one line (Section 2.3.1).
#	Denotes a device or file # name, or is used as an output format effector (Chapter 7 and Section 10.4).
,	Output format effector and list terminator (Section 3.3).
;	Output format effector (Section 3.3).
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).
[ ]	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	Decimal Value	ASCII Character	RSTS Usage
64	@		96	`	
65	A		97	a	
66	B		98	b	
67	C		99	c	
68	D		100	d	
69	E		101	e	
70	F		102	f	
71	G		103	g	
72	H		104	h	
73	I		105	i	
74	J		106	j	
75	K		107	k	
76	L		108	l	
77	M		109	m	
78	N		110	n	
79	O		111	o	
80	P		112	p	
81	Q		113	q	
82	R		114	r	
83	S		115	s	
84	T		116	t	
85	U		117	u	
86	V		118	v	
87	W		119	w	
88	X		120	x	
89	Y		121	y	
90	Z		122	z	
91	[		123	{	
92	\	Backslash	124	|	Vertical Line
93	]		125	}	
94	^ or ↑		126	~	Tilde
95	_ or ←		127	DEL	RUBOUT

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$_{1\emptyset}$	DECØ29	DECØ26	14Ø1
{	123	12 Ø	12 Ø	UNUSED
}	125	11 Ø	11 Ø	UNUSED
SPACE	32	NONE	NONE	NONE
!	33	12 8 7	12 8 7	11 Ø
"	34	8 7	Ø 8 5	Ø 8 2
#	35	8 3	Ø 8 6	8 3
$	36	11 8 3	11 8 3	11 8 3
%	37	Ø 8 4	Ø 8 7	Ø 8 4
&	38	12	11 8 7	12
'	39	8 5	8 6	12 8 4
(	4Ø	12 8 5	Ø 8 4	8 7
)	41	11 8 5	12 8 4	Ø 8 7
*	42	11 8 4	11 8 4	11 8 4
+	43	12 8 6	12	Ø 8 5
,	44	Ø 8 3	Ø 8 3	Ø 8 3
-	45	11	11	11
.	46	12 8 3	12 8 3	12 8 3
/	47	Ø 1	Ø 1	Ø 1
Ø	48	Ø	Ø	Ø
1	49	1	1	1
2	5Ø	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	Ø 8 2	11 8 6
<	6Ø	12 8 4	12 8 6	12 8 6
=	61	8 6	8 3	11 8 7
>	62	Ø 8 6	11 8 6	8 6
?	63	Ø 8 7	12 8 2	12 Ø

CHARACTER	ASCII$_{1\emptyset}$	DECØ29	DECØ26	14Ø1
@	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	7Ø	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	8Ø	11 7	11 7	11 7
Q	81	11 8	11 8	11 8
R	82	11 9	11 9	11 9
S	83	Ø 2	Ø 2	Ø 2
T	84	Ø 3	Ø 3	Ø 3
U	85	Ø 4	Ø 4	Ø 4
V	86	Ø 5	Ø 5	Ø 5
W	87	Ø 6	Ø 6	Ø 6
X	88	Ø 7	Ø 7	Ø 7
Y	89	Ø 8	Ø 8	Ø 8
Z	9Ø	Ø 9	Ø 9	Ø 9
[	91	12 8 2	11 8 5	12 8 5
\	92	Ø 8 2	8 7	Ø 8 6
]	93	11 8 2	12 8 5	11 8 5
↑ or ^	94	11 8 7	8 5	unused
← or _	95	Ø 8 5	8 2	12 8 7

EOF is 12-11-Ø-1 punch or a 12-11-Ø-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

INDEX

RSTS-11 User's Guide
DEC-11-ORSUA-A-D
September, 1972

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability and readability.

Did you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Please state your position. ______________________ Date: __________

Name: ______________________ Organization: ______________________

Street: ______________________ Department: ______________________

City: ______________ State: ______________ Zip or Country __________

------------------------- 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

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

digital

DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS 01754