$1.00

DIGITAL EQUIPMENT CORPORATION

education

EDUSYSTEM 10 System User's Guide

digital

working at the computer

EduSystem 10 is a two-part system—a computer which is housed in a small cabinet and a typewriter-like device called a terminal. Each of these units (which are connected by wire) has a specific purpose. The computer does all the actual calculations and processing; it is the "brain." The terminal does all the communicating between the computer and the person using it; it is the "eyes and ears."

The computer always acts in response to a command given to it by the user. Commands are given to the computer by typing them at the terminal. Typing at the terminal is just like typing at an electric typewriter, with one important difference: the characters which are typed are transmitted to the computer as well as being printed. When a complete command has been typed in, the computer processes it and makes its response by typing a message at the terminal. This message might be some decimal values which are the results of calculations the computer was commanded to perform. It might also be an "error message" explaining that the computer could not understand a command. Or it might be one of a variety of other responses which are described later in this user's guide.

This is really all there is to working at the computer. The user types in commands which tell the computer to perform specific actions. Generally, these actions include the typing out of some sort of information. The computer performs the specified actions, types out the requested information and waits for more instructions. The user and the computer simply talk back and forth.

The computer does not, however, respond to just any command. It knows only a limited vocabulary and requires that commands be typed according to certain rules and conventions. These rules make up the computer's "language." The language of EduSystem 10 is BASIC which is both easy to learn and very powerful. The remaining sections of this guide describe the BASIC language.

The computer also requires that its instructions be quite explicit. The tasks which it is to do must be spelled out exactly. Lengthy procedures must be broken down into steps.

Thus, although the actual operation of the computer is very simple, the task it may be given can be quite complex. As the user gains more experience, he may take advantage of some of the advanced capabilities of the BASIC language. These may be used to develop more and more complex sequences of commands. These commands may then be organized into more and more sophisticated programs.

A number of rules and conventions should be remembered whenever the computer is used. The terminal does not have lower case letters; therefore, all commands, and the computer responses, always use capitals. The RETURN key is very important when using the computer. Not only does it return the carriage, it also signals the computer that a command line is complete. The computer does not respond to a command until the RETURN key is typed. It must be used, therefore, at the end of every typed-in line.

Two special keys may be used to correct typing mistakes. The ALT MODE key cancels the line being typed, allowing the user to start over. The back arrow (←) key, which is SHIFT/O on the keyboard, may be used to erase just the previous character. More than one back arrow deletes more than one character. ALT MODE and back arrow may only be used to correct a command before the RETURN key is typed. Once RETURN is typed, the computer begins to execute the command, even if it is erroneous. The S key may be used in this situation to cancel the computer's response.

RETURN key	Signal for the computer to act on the command.
ALT MODE key	Cancel the line being typed.
Back Arrow (←)	Delete the character just typed.
S	Cancel the response being made by the computer.

calculating with the computer

*Color indicates computer printout

EduSystem 10 may be used to perform calculations of most any complexity. The user types in the algebraic expression whose value is to be calculated, the computer types back the value. For example:

PRINT 2+7—4
5

The PRINT command tells the computer to calculate the value of the expression and then print it out. Whenever the computer is to print out the results of calculations, the PRINT command must be used.

Expressions may be made up of any decimal numbers and the five regular algebraic operations; addition, subtraction, multiplication, division, and exponentiation. Multiplication is indicated by an asterisk (*), division by a slash (/), and exponentiation by an up-arrow (↑). Numbers may have up to seven significant digits. Additional digits are ignored by the computer. The following examples show the use of various expressions.

PRINT 3.14159*4↑2 **Area of a circle of radius 4**
50.26544

PRINT 17/12 **Approximation of $\sqrt{2}$**
1.416667

The computer always types answers in decimal form. If the answer is too big or too small to be typed in its normal form, it is typed in scientific notation; the number followed by the power of 10 by which it should be multiplied.

PRINT 8↑8
1.677722E+07

PRINT 1/8↑8
5.960464E—08

As expressions become more complex, the order in which they are evaluated becomes important. To prevent confusion, the computer always evaluates according to a fixed formula. Exponentiation is done first, then multiplication and division, then addition and subtraction. Operations of equal priority are performed from left to right. Thus, 1/8 ↑ 8 is evaluated as one over 8 to the eighth, not one-eighth raised to the 8th power. But, 1/2*3 is evaluated as one-half times three, not one over the quantity two times three. Parentheses may and should be used to indicate the desired order of evaluation. Operations within parentheses are always performed first. For example:

PRINT 5/9*(98.6—32) **Convert 98.6 to Centigrade**
37

Standard functions may also be used in calculations. Functions are indicated by typing the three letter name of the function, then the argument enclosed in parentheses. The value of the function is the result of applying the function to the specified argument. For example, the value of SQR(9) is 3, the square root of 9. The functions which may be used are:

SQR	Square root
SIN	Sine—argument in radians
COS	Cosine—argument in radians
TAN	Tangent—argument in radians
ATN	Arctangent—argument in radians
EXP	Exponential
LOG	Natural log
ABS	Absolute value
INT	Integer part
SGN	Sign
RND	Random numbers

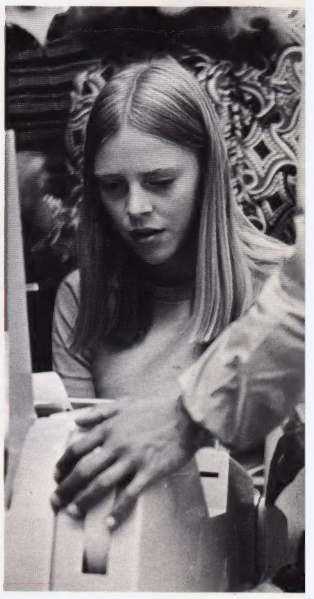
Functions may be used anywhere within an expression. The argument of the function may itself be an expression as in the examples below.

PRINT (—5+SQR(5↑2—4*2—12))/(2*2)
1.5 **A solution of $f(x)=2x^2+5x-12$**

PRINT SQR(5↑2+7↑2) **Length of hypotenuse when sides are 5 and 7**
8.602325

Most of the functions listed above are familiar from algebra and trigonometry. The others are unique to computer language. The integer function (INT) is used to convert a decimal number to an integer. The INT function produces the greatest integer which is less than the argument. The SGN function is used to determine the sign of a number. The value of the function is 1 if the argument is positive, —1 if it is negative, zero if it is zero. RND is used to obtain a random number between zero and one. Each time RND is used, RND returns a different number. Thus, the value of the RND function is in no way dependent on its argument. It must, however, always have an argument when it is called. This "dummy argument" is required by the BASIC language.

more complex calculations



All of the calculations described in the previous section were single-step problems. In reality, most problems involve a series of steps where the results of each step are used in the next step. To do these sorts of calculations, the computer must be able to remember the intermediate results. Program variables are used for this purpose. Just as in algebra, a variable is a symbol which stands for a particular value. The actual value that it has is the last value that has been assigned to it. The variable names themselves may be any of the capital letters, or a letter followed by a single digit.

The command LET is used to give a value to a variable. A variable may be set equal to a constant or an arithmetic expression. For example:

```
LET P1=3.14159
```

The computer does not type anything in response to a LET command. It merely remembers the indicated value of the variable. This value may then be used in PRINT statements to type out information.

```
PRINT P1*4↑2
 50.26544
```

By using LET commands and PRINT commands, the computer may be used to solve a multi-step problem, the solution of a quadratic equation, for example.

```
LET A=2
LET B=5
LET C=−12
```

The values of A, B, and C for the function $f(x)=2x^2+5x-12$ are loaded into the computer. They may then be used to calculate the discriminant.

```
LET D=SQR(B↑2−4*A*C)
```

The discriminant is an intermediate value. Since it is not of interest, it is not typed out. It is, however, used to calculate the actual roots.

```
LET R1=(−B−D)/(2*A)
LET R2=(−B+D)/(2*A)
```

These two roots may then be printed out.

```
PRINT R1, R2
 −4          1.5
```

Note that PRINT statements may be used to type out more than one value. In this case the values typed out are not expressions, but the current values of the program variables R1 and R2 which contain the roots of the equation.

In a simple step-by-step fashion, the computer has been used to find the roots of the quadratic $2x^2+5x-12$. By changing the values of A, B, and C, the same process could be repeated for a different equation. (It might be prudent in this case to print out the value of b^2-4ac to make sure it is positive.)

NOTE: In all of the above examples, the word LET could have been omitted.

```
D=SQR(B↑2−4*A*C)
```

is completely equivalent to:

```
LET D=SQR(B↑2−4*A*C)
```

writing a program

So far, the computer has been used to simply step thro the solution of a problem. The two commands LET and PRINT were used to perform calculations, remember intermediate results, and print out the final solution. This step-by-step method of operation does not, however, begin to use the full power and flexibility of EduSystem 10. To do more complex work, it is usually best to develop a "stored program."

In the step-by-step, or calculator, mode of operation, the computer receives its commands one at a time. As soon as each command is typed in, it is executed. The computer then waits for the next command. In the case of LET commands, the computer remembers the resulting value. But in all cases, the command itself is "forgotten" by the computer as soon as it is executed. If the same calculation is to be performed twice on two different sets of data, the whole sequence of commands must be typed in each time.

The stored program mode of operation allows the computer to remember a sequence of commands as well as the intermediate results. Instead of executing each command as it is typed in, the computer stores them away in its memory. When the whole sequence of commands has been entered, the computer may be commanded to execute them all at once.

The commands which make up a stored program look exactly like the commands used to calculate except they have a statement number in front of them. example:

```
10 LET A=2
```

The statement number (which may be any integer from 1 to 2045) indicates that the computer is not to perform the command yet. All it does is to store it away. Additional stored commands may then be entered until the entire program is complete.

When all commands of a program have been entered, the entire program may be executed, or "run." The computer does not pause after it completes an individual command. It starts with the command have the lowest statement number and proceeds through each statement in the program, in order of statement number. The following program could be used to solve a quadratic equation:

```
10 LET A=2
20 LET B=5
30 LET C=-12
40 LET D=SQR(B↑2-4*A*C)
50 LET R1=(-B-D)/(2*A)
60 LET R2=(-B+D)/(2*A)
70 PRINT R1, R2
80 END
```

Once it has been entered, a stored program statement remains stored until it is changed or deleted. (See the section on Modifying a Program.) If a new program is to be entered into the computer, all steps of a previ program must be deleted first. The command SCR (m ing scratch) is used to clear out the computer's memory. SCR should always be typed at the beginning of a session to assure that no old programs remain.

SCR	Delete all stored commands from the computer's memory

learning the basic language

The previous sections have already used many elements of the BASIC language. The PRINT and LET statements are two of the BASIC commands. Constants, expressions, variables, and functions are also important elements of BASIC. Little additional knowledge of the language is needed to write meaningful programs. There are, however, many additional and very useful BASIC capabilities. These are described in the following pages.

The description of BASIC in this guide will be most helpful to the reader who already has some familiarity with computers and with BASIC. It describes the features of BASIC which are available with EduSystem 10. For the user, or student, who is just getting started with computers, EduSystem 10 includes a variety of tutorial introductions to BASIC. These books also integrate the use of the computer into the standard math and science curriculum.

texts and curriculum materials

Basic BASIC, Hayden Publishing.

Computer Methods in Mathematics, Addison-Wesley.

Teach Yourself BASIC, Volume I, Tecnica.

Teach Yourself BASIC, Volume II, Tecnica.

Fundamentals of Digital Computers, Howard W. Sams.

Problem Solving with a Computer, Entelek.

Computer Assisted Mathematics Program (CAMP), Scott, Foresman.

- *CAMP First Course,* text and teacher guide.
- *CAMP Second Course,* text and teacher guide.
- *CAMP Algebra,* text and teacher guide.
- *CAMP Geometry,* text and teacher guide.
- *CAMP Intermediate Math,* text and teacher guide.

Introduction to Computer Science, Scott, Foresman.

Introduction to Programming, DEC.

Programming Languages, DEC.

A FOCAL Primer, Cornell Press.

FOCAL-8 Demonstration Programs, DEC.

BASIC Applications Programs, DEC.

BASIC Simulation Programs, DEC.
- Volume 1 (Mathematics)
- Volume 2 (Physics, Chemistry, Biology, Ecology)
- Volume 3 (Social Studies, Business, Teacher Aids, Miscellaneous)

FOCAL Physics Experiments, DEC.

commands

THE LET COMMAND

The LET command is used to set a variable equal to an expression. It is the standard way to do computations in BASIC. The general form of the LET command is:

line number LET variable=expression.

All LET commands must be followed by a single variable, which may be subscripted, an equal sign (=), and a valid expression, variable, or constant. The LET command itself may be omitted. The following are examples of the LET command.

```
100 LET R=F
110 P=3.141592
120 LET R1=(-B+SQR(B↑2-4*A*C))/(2*A)
121 LET R2=(-B-SQR(B↑2-4*A*C))/(2*A)
130 LET L=L+1
```

The LET command is not strictly an equality. LET means "evaluate the expression to the right of the equal sign and assign this value to the variable on the left." Thus, the statement L=L+1 is a meaningful command. It means "set L equal to a value one greater than it was before."

THE PRINT COMMAND

The PRINT command is used to type out information, such as the value of a constant, variable, or expression. The value is typed as a decimal number accurate to seven significant digits. Very large or very small values are printed in scientific notation. The following are examples of PRINT commands:

```
100 PRINT 2+2
149 PRINT Z4
220 PRINT A(9)
333 PRINT
987 PRINT 3.14159*R↑2
```

The PRINT command may also be used to type out text which is enclosed in quotation marks. A line may contain just text or it may contain both text and a variable. For example:

```
210 PRINT "ROOTS ARE COMPLEX"
294 PRINT "PROGRAM TO CONVERT FARENHEIT"
333 PRINT X "MILLIMETERS"
382 PRINT "THE AVERAGE IS" A
942 PRINT "RATE IS" Q "MILES PER HOUR"
```

The PRINT command allows more than one value to be typed out on a line. Two special characters, the comma (,) and the semi-colon (;) are used for this purpose. Variables separated by semi-colons are printed out as close together as possible. The comma provides a more sophisticated formatting capability. Variables separated by commas are aligned at one of five "print zones" across the page. If there is no more room on the line, the value is printed on the next line. Examples of PRINT using commas or semi-colons:

```
123 PRINT A; B; C; D
144 PRINT A, B, C
182 PRINT A,,C
223 PRINT A, B, C, D, E, F
```

All of the capabilities described above may be combined in a single PRINT statement. Commas may be used to align text within a particular print zone. Several values, plus text, may be combined in a single PRINT statement.

READ, DATA, RESTORE, INPUT

BASIC programs may use either of two ways of providing a program with its initial data. This data may be entered in a list as part of the program itself or it may be typed in from the keyboard while the program is running. If the data is entered as part of the program, a DATA statement is used. A DATA statement may contain any number of values as long as they fit on a single line. All values must be in decimal or scientific notation format. For example:

234 DATA 2, 3, −4, .2654, 2.8E−4, −.012

If the program's input data will not fit in a single line, additional DATA statements may be used. If a program has multiple DATA statements, the order in which they are used is determined by the order of their line numbers.

The data in a program's DATA statements is not used directly by the program. These values must first be read in and stored in program variables. The READ statement is used to fetch a data item and assign its value to a variable. For example:

100 READ A,B

takes the next two data items from the DATA statement and sets A equal to the first and B equal to the second. One or more variables may be initialized with a single READ statement. The first READ statement reads the first data item from the first DATA statement in the program. Successive READ's read successive items from this DATA statement or, if there are no more, the first item from the next DATA statement. Since READ and DATA operate in conjunction with each other, the DATA statements must be arranged in the proper order. It is good practice to group all DATA statements together at the end of a program.

Since each READ goes on to the next data item, it is not possible to read the same item twice. It is, however, possible to re-read the entire data list. The RESTORE statement resets the program so that the next READ statement will read the first value in the first DATA statement.

299 RESTORE

READ and DATA statements require that all initial data be stored within the program before it is executed. The INPUT command allows data to be typed directly into the program while it is running. For example:

194 INPUT A, B, C

When the computer executes this statement, it prints a question mark on the terminal and waits for the user to type in these values, separated by commas. The variables B, and C are then set equal to these typed-in values. PRINT may be used with INPUT to request specific values from the user. For example,

190 PRINT "ENTER CO-EFFICIENTS";
195 INPUT A, B, C

One or more variables may be initialized by a single INPUT statement.

END STOP REM

All BASIC programs should conclude with an END statement.* The END statement has a higher line number than any of the other statements in the program. Unlike commands such as LET and PRINT, END does not command the computer to do anything. It merely acts as a signal that the entire program has been executed. The END statement does not take any arguments. For example:

999 END

The END statement must always come at the end of the program. A STOP statement may be used to terminate a program before the end. A program may contain any number of STOP statements. Like END, STOP is used without any other information on the line.

55 STOP

REM (Remarks) statements may be used anywhere in a program to improve its readability. REM statements are always ignored when a program is run. They simply contain comments or remarks about the program itself.

765 REM CHECK THAT DISCRIMINANT IS NOT NEGATIVE

*EduSystem 10 does not require that the END statement be used. It should, however, be included to retain compatability with the other EduSystems.

commands

GOTO IF

Unless otherwise instructed, the computer will execute a program starting at the statement with the lowest line number and proceeding through to the statement with the highest line number, the END statement. Each statement (with the exception of DATA and REM statements which are not executed) is executed exactly once. This implied order of execution may be changed by means of a GOTO command, which explicitly states the line number to be executed next. For example:

```
301 GOTO 201
```

If the line number specified in the GOTO statement is a previous line number, the GOTO will cause previous statements to be re-executed. If the GOTO specifies a subsequent line number, statements are skipped over without being executed.

An IF statement may be used to GOTO another statement if a specified condition is true. If the condition is not true, the GOTO is not performed and the next statement in the program is executed. The general form of the IF statement (often called a conditional GOTO) is:

IF relation GOTO line number
or IF relation THEN line number

Both forms of the command have the same meaning. The "relations" are familiar algebraic equalities and inequalities: >, =, <, >=, <=, <>. They may be used to compare any two expressions. If the whole relational is true, the GOTO is performed. The following are examples of IF statements:

```
200 IF A=0 THEN 50
225 IF B↑2-4*A*C<0 GOTO 400
333 IF Q>2.8 THEN 444
```

FOR NEXT

It is possible, using IF and GOTO commands, to repeat a section of a program a specified number of times. This programming technique, called looping, is used so frequently, however, that BASIC has a special set of commands, FOR and NEXT, for just this purpose. The FOR statement indicates that the statements following it are to be executed a certain number of times. The NEXT statement marks the end of the group of statements which are to be executed repetitively. Thus, FOR and NEXT bracket a loop. The general form of FOR and NEXT are:

FOR variable=expr TO expr STEP expr
NEXT variable

When a FOR statement is first executed, the variable is set equal to the value immediately after the equal sign. The subsequent statements are executed up to the NEXT statement. The value following STEP is then added to the variable. If the result is greater than the value following the TO, the loop is ended and the statement following the NEXT is executed. If not, the loop is repeated starting with the statement after the FOR statement. Eventually, the variable will exceed the value following the TO. At this time, execution of the program loop will be ended. For example, the following two BASIC programs do exactly the same thing.

```
10 FOR I=1 TO 10 STEP 1
20 PRINT SQR(I)
30 NEXT I
40 END
```

```
10 LET I=1
20 PRINT SQR(I)
30 LET I=I+1
40 IF I<=10 THEN 20
50 END
```

The step value may be a number other than 1. It may be minus if the initial value of the variable is greater than the final value. If the step value is 1, the "STEP 1" may be omitted from the FOR command. The following are all valid FOR commands:

```
100 FOR J=1 TO 10 STEP 4
120 FOR I=-20 TO 20 STEP .2
130 FOR K=1 TO 10
140 FOR M=10 TO 1 STEP -1
```

It is also possible to "nest" FOR loops one within another providing that the inner loop is completely contained in the outer loop.

GOSUB RETURN

Programs frequently contain sequences of code which are repeated, either exactly or very closely. BASIC's sub-routine capability allows these command sequences, called sub-routines, to be written only once, but used throughout the program.

Instead of being included at the place in the program where they will be used, a sub-routine sequence is moved to another part of the program. The end of the program, just before the END statement, is a convenient place to put any sub-routines. The last statement of a sub-routine must be a RETURN statement. For example:

899 RETURN

At the place or places in the main program where the sub-routine statements are to be executed, a GOSUB instruction is used. Included in the GOSUB statement is the line number of the first line of the sub-routine. For example:

200 GOSUB 800

GOSUB is like GOTO in that it changes the order of execution of the program. The difference is that at the end of the sub-routine, as indicated by the RETURN statement, the next statement to be executed is the one following the GOSUB. The GOSUB goes to the sub-routine, then automatically returns to the statement after the GOSUB. A program may have many GOSUB's, each of which go to the same sub-routine. In each case, the program executes exactly as though the sub-routine statements were inserted in the program in place of the GOSUB.

If sub-routines are used, it is important to make sure that they are kept distinct from the main program. This is normally done by placing them after the main program. In addition, the last statement in the main program must be a STOP or a GOTO.

DIM DEF

Programs variables may have one or two subscripts. For example:

A(1, 1) B(I, J) X(1) A1(4)

If program variables are to be subscripted, the maximum size of the subscripts should be declared in a DIM statement. For example:

5 DIM A(4,4), B(3,3), X(4), A1(5)

specifies four subscripted variables, two with double subscripts, two with single subscripts. If no DIM statement appears in the program, subscripts are assumed to be of maximum size 10.

Programs may use any of the standard functions or may define their own. A DEF statement may be used to define such a function. For example:

10 DEF FNA(F)=5/9*(F—32)

defines a function FNA which may then be used elsewhere in the program. Defined functions must have three letter names, starting with FN. Functions may have a single argument only. Each EduSystem 10 program may have only one DEF statement.

running a stored program

As each step of a program is entered, it is simply stored away in the computer. When all steps have been entered, the program may be executed, or run, by typing the RUN command.

RUN	Execute the program.

RUN commands the computer to start executing the program at the statement having the lowest statement number. It continues to execute statements until a STOP or END command is encountered. STOP and END both stop the execution of the program. The computer types "READY" to indicate that it is ready to receive a new command.

If the program does not run properly, or contains an infinite loop and, hence, will never stop, it may be terminated by typing the S key.

S	Stop program execution.

modifying the program

Unless a program is very simple, it probably won't run properly the first time. Programs errors, or "bugs," must be found and corrected before the program does what it was intended to do. Generally, the first step in finding a program error is to obtain an up-to-date listing of the program. The LIST command may be used for this purpose.

LIST	List the program.
LIST n	List the program starting at line number n.

The S command may be used to stop a listing.

Individual lines may be changed by simply typing them in again. Whenever a line is entered, it replaces any existing line which has the same line number. New lines may be inserted anywhere in the program by giving them line numbers which are between two other existing line numbers. An existing line may be deleted by typing just its line number. Using these editting capabilities, the program may be modified and re-run until it works properly.

saving and reloading programs

Once a program has been typed in, it may be saved on paper tape so that it may be reloaded quickly. To save the program, first type LIST, but do not press the RETURN key. Now turn the paper tape punch on. Type the HERE IS key to produce some leader tape. Then press the RETURN key. The program will be punched and listed. When the punching is complete, type the HERE IS key again to produce some trailer tape. Then turn the punch off and remove the tape.

Programs punched out on paper tape may be loaded through the paper tape reader. Type SCR to delete any previous program. Then put the tape in the reader and turn the reader on. The program will be listed as it is being loaded into the computer.

advanced capabilities

EduSystem 10 allows more than one command to be typed on a single line. Commands after the first begin with a back slash character (\ which is SHIFT/L on the keyboard) and must not have a line number. Typing multiple commands per line is especially useful in the direct mode. A table of the first 10 square roots could, for example, be generated by the following single line:

```
FOR I=1 TO 10\PRINT SQR(I),\NEXT I
```

Although they are not commonly used, other BASIC commands are available in the direct mode. For example, a direct GOTO command may be used to start a program at a point other than the beginning.

The ability to put multiple statements on a line is also useful when writing programs. The program is often more understandable when statements, such as a series of LET's, are grouped into a single line. For example:

```
100 A=1 \ B=4 \ C=6
```

This capability may also be used when the program to be written is too big for the computer to hold. Commands take less storage room in the computer when they are grouped as a single statement.

EduSystem 10 also allows variables to be entered as input from the terminal. This capability might be used as follows:

```
100 Y=1\N=2
110 PRINT "TRY AGAIN (ANSWER Y OR N)";
120 INPUT A
130 IF A=Y THEN 20
140 STOP
```

error messages

The computer checks all commands before executing them. If for some reason, it cannot execute the command, it reports the fact by typing one of the following messages. The number of the line in which the error was found is also typed out.

SYNTAX ERROR	The command does not correspond to the language syntax. Common examples of syntax errors are misspelled commands, unmatched parens, and other typographical errors.
FUNCTION ERROR	A function was used which was deleted when the system was loaded and hence is not available. A DEF statement will produce this error if the DEF capacity has been deleted.
TOO-BIG ERROR	The combination of program and variables exceeds the capacity of the computer. Reducing one or the other may help. If the program has undergone extensive revision, try punching it out, typing SCRATCH, and reloading.
SUBSCRIPT ERROR	A subscript has been used which is outside the bounds defined in the DIM statement.
LINENO ERROR	A GOTO, GOSUB, or IF references a nonexistent line.
FOR ERROR	FOR loops are nested too deeply.
NEXT ERROR	FOR and NEXT statements are not properly paired.
GOSUB ERROR	Subroutines are nested too deeply.
RETURN ERROR	GOSUB and RETURN statements are not properly paired.
DATA ERROR	There are no more items in the data list.
ARGUMENT ERROR	A function has been given an illegal argument, for example SQR(—1).

initial installation

When EduSystem 10 is first installed, it is loaded with a special software program; the BASIC language processor. Once this software is loaded, it need not be reloaded.

1. Plug the EduSystem 10 computer into a 3-prong, 60 cycle electrical outlet. Plug the terminal into the outlet provided at the back of the computer cabinet. Turn the key lock on the front of the computer to the POWER ON position, but not all the way to LOCK. Set all switches to the left of the LOAD ADDR switch to their down position.

2. If the computer includes a "Diode Loader", this step may be omitted. Otherwise, perform the following set of switch manipulations. In each step, there are 12 figures which correspond to the 12 switches which are labeled the Switch Register (SR) on the front of the computer. The ● symbol indicates that the corresponding switch should be set to its "up" position.
The ○ symbol means that the corresponding switch should be set to its "down" position.

Set SR to: ●●● ●●● ●○● ●●○ ; then press LOAD ADDR
Set SR to: ●●○ ○○○ ○●● ○●○ ; then lift DEP
Set SR to: ●●○ ○○○ ○●● ○○● ; then lift DEP
Set SR to: ●○● ○●● ●○● ●●● ; then lift DEP
Set SR to: ●●○ ○○○ ○●● ●●○ ; then lift DEP
Set SR to: ●●● ○○● ○○○ ●●○ ; then lift DEP
Set SR to: ●●● ○○○ ○○○ ●●○ ; then lift DEP
Set SR to: ●●● ●○● ○○● ○○○ ; then lift DEP
Set SR to: ●○● ○●● ●○● ●●● ; then lift DEP
Set SR to: ●●● ○○○ ○○○ ●●○ ; then lift DEP
Set SR to: ●●○ ○○○ ○●● ○○● ; then lift DEP
Set SR to: ●○● ○●● ●●○ ●●● ; then lift DEP
Set SR to: ●●○ ○○○ ○●● ●○○ ; then lift DEP
Set SR to: ●●● ●○○ ○●○ ○○○ ; then lift DEP
Set SR to: ○●● ●●● ●●● ●●○ ; then lift DEP
Set SR to: ○●● ○●● ●●● ●●○ ; then lift DEP
Set SR to: ●○● ○●● ●○● ●●○ ; then lift DEP

3. Place the tape labeled EDUSYSTEM 10 in the terminal's paper tape reader. At the front of this tape is about 8" of leader tape which has just a single row of holes punched. Put the tape in the reader at this point.

4. Set the SR to ●●● ●●● ●○● ●●○ ; then press LOAD ADDR. Then press the START switch. (Press the CLEAR switch, then the CONT switch if the computer is a PDP-8/E). Push the paper tape reader switch to the START position. The tape will read in. If it stops before the end of the tape, an error has occured. Repeat steps 2, 3, and 4.

5. When the tape has read in properly, a message is typed on the terminal. At this time it is possible to delete any functions which will not be used. Deleting functions increases the size of the BASIC program which may be accommodated. If all functions are deleted, the maximum program size is roughly 60 lines. If all functions are retained, the maximum program size is roughly 30 lines.

6. After the functions have been selected, indicate whether subscripts will begin at zero or one. Most BASIC programs do not use the zeroth element of an array. If this will be the case, setting subscripts to start at one allows larger programs to be run.

7. EduSystem 10 is now ready for use. Turn the key lock to LOCK and remove the key to prevent the system from being disturbed accidentally.

turning the system off

1. Type the S command at the terminal to stop any program which might be running.
2. Turn the key lock to the POWER position, then press the STOP switch. (If the computer is a PDP-8/E, press the HALT key, then raise it again.)
3. Turn the key lock to OFF.

restarting the system

1. Turn the key lock to POWER.
2. Set the SR to ↓↓● ↓↓↓ ↓↓↓ ↓↓↓; then press LOAD ADDR. Then press START. (Press the CLEAR switch, then the CONT switch if the computer is a PDP-8/E.)
3. EduSystem 10 is now ready for use. Turn the key lock to LOCK and remove the key to prevent the system from being disturbed accidentally.

reloading the functions

1. Type the S command at the terminal to stop any program which might be running.
2. Turn the key lock to the POWER position, then press the STOP switch. (If the computer is a PDP-8/E, press the HALT key, then raise it again.)
3. Follow the procedure for Initial Setup, starting at step #3.

However, instead of using the EDUSYSTEM 10 tape, use the shorter tape labeled EDUSYSTEM 10 FUNCTIONS.

basic command summary

BASIC Statements

LET	Assign a value to a variable. LET is optional.
PRINT	Print out the indicated information.
READ	Initialize variables to value from the data list.
DATA	Provide initial data for a program.
GOTO	Change order of program execution.
IF GOTO IF THEN	Conditionally change order of program execution.
FOR TO STEP	Set up a program loop.
NEXT	End a program loop.
GOSUB	Go to a sub-routine.
RETURN	Return from a sub-routine.
INPUT	Get initial values from the terminal.
REM	Insert a program comment.
RESTORE	Restore the data list.
DEF	Define a function.
STOP	Stop program execution.
END	End a program.
DIM	Define subscripted variables.

BASIC Functions

SIN (X)	Sine of x (x in radians)
COS (X)	Cosine of x (x in radians)
TAN (X)	Tangent of x (x in radians)
ATN (X)	Arctangent of x (x in radians)
EXP (X)	e^x (e=2.718282)
LOG (X)	Natural log of x
ABS (X)	Absolute value of x
SQR (X)	Square root of x
RND (X)	Random number

BASIC Edit and Control Commands

LIST	List all stored program statements.
RUN	Run the currently stored program.
SCR	Delete the currently stored program.
S	Stop execution of a program.

Printed in U.S.A. .081C - 00371 - 1[illegible]-03

digital

DIGITAL EQUIPMENT CORPORATION, Maynard, Massachusetts, Telephone: (617) 897-5111 • ARIZONA, Phoenix • CALIFORNIA, Anaheim, Los Angeles, Oakland, Palo Alto • COLORADO, Denver • CONNECTICUT, Meriden • DISTRICT OF COLUMBIA, Washington, (College Park, Md.) • FLORIDA, Orlando • GEORGIA, Atlanta • ILLINOIS, Chicago • INDIANA, Indianapolis • MASSACHUSETTS, Cambridge and Waltham • MICHIGAN, Ann Arbor • MINNESOTA, Minneapolis • MISSOURI, St. Louis • NEW JERSEY, Parsippany and Princeton • NEW MEXICO, Albuquerque • NEW YORK, Centereach (L.I.), New York City, (Englewood, N.J.), and Rochester • NORTH CAROLINA, Durham/Chapel Hill • OHIO, Cleveland and Dayton • OREGON, Portland • PENNSYLVANIA, Philadelphia and Pittsburgh • TENNESSEE, Knoxville • TEXAS, Dallas and Houston • UTAH, Salt Lake City • WASHINGTON, Seattle • AUSTRALIA, Brisbane, Melbourne, Perth and Sydney • CANADA, Edmonton, Alberta; Vancouver, British Columbia; Carleton Place, Ottawa and Toronto, Ontario; and Montreal, Quebec • ARGENTINA, Buenos Aires • BELGIUM, Brussels • ENGLAND, London, Manchester and Reading • FRANCE, Paris • GERMANY, Cologne, Hanover, Frankfurt and Munich • HOLLAND, The Hague • ITALY, Milan • JAPAN, Tokyo • SWEDEN, Stockholm • SWITZERLAND, Geneva and Zurich.