PCM-12A

4K Basic

PC/M, Inc.

San Ramon, CA

PCM

4K BASIC

PROGRAMMER'S MANUAL

PCM, Inc.
San Ramon, CA

December, 1975
Document #12890

TABLE of CONTENTS

SECTION 1.0

INTRODUCTION

CINET-BASIC is a binary-format program originally written for DEC's PDP-8 computer by Bud Pembroke and David Gillette of the Computer Instruction Network, Salem, Oregon. The program is fully compatible with the PCM-12 computer manufactured by PCM, Inc., San Ramon, California. While not as powerful as the other BASIC language interpreters available from DEC for the PDP-8 series computers, CINET-BASIC is nonetheless a powerful programming tool in the hands of an experienced pregrammer. For the beginner, it is perhaps easier to learn than most forms of BASIC, due to the abbreviated list of available instructions. The interpreter requires a PCM-12 with 4096 words of memory and a Model 33 or 35 Teletype or CRT terminal as a minimum system configuration. The Teletype machine will allow use of paper tape; a CRT terminal used for input/output will require use of a separate paper tape reader for program entry, or optionally, an audio cassette recorder with the cassette form of the interpreter. No listing of the interpreter is available. The CINET-BASIC interpreter is available to members of the Digital Equipment Corporation User's Society (DECUS) in paper-tape form. Ordered from DECUS, CINET-BASIC comes with a PDP-8 oriented write-up. The DECUS catalog number is #8-159.

This BASIC-language interpreter has been successfully checked out by PCM, Inc. on the PCM-12 computer. However, PCM makes no representation of any kind as to the accuracy, reliability or performance of this software, and assumes no responsibility for the use of CINET-BASIC in the user's equipment or system. PCM reserves the right to change, modify, delete or add to the information given in this manual, and to the binary paper tape, at any time without notice.

NOTE: DEC, PDP, FOCAL and DECUS are registered trademarks of Digital Equipment Corporation, Maynard, MA. Teletype is a registered trademark of the Teletype Corporation, Skokie, IL.

2.0 HOW TO LOAD AND START CINET-BASIC ON THE PCM-12

To Load

1. Lift RESET on the PCM-12.
2. Set the SWITCH REGISTER for the desired reader (for example set Switch Ø, only, for the Teletype paper-tape reader).
3. Set the Rotary Switch to the AC position.
4. Place the CINET-BASIC binary tape in the reader. Be certain that leader/trailer code (channel 8, only, punched) is over the read head.
5. Lift BIN BOOT.
6. Enable the reader (for example, on the Teletype reader, set the reader control switch to START).

The CINET-BASIC tape is in three sections. It will stop after each section to show a check-sum result in the Display lamps. If the preceding section was read in correctly the CPU Accumulator will be all zero's, as indicated by the Display lamps, when the tape stops. If it is other than all zero's, it is necessary to read the preceding section (only) in again. After each section is read in properly, simply lift RESET and BIN BOOT to start reading the next section.

The third section of the tape contains the extended functions: SIN, COS, ATN, LOG and EXP. They are loaded at the option of the user. They consume about 800 locations of memory space that can otherwise be used to hold additional user program instructions. If you will not need the extended functions, simply omit loading section three of the tape. If you decide to load section three later, be sure to type "ERASE ALL" prior to loading section three.

To Start

1. Set 0200_8 in the SWITCH REGISTER.
2. Lift RESET
3. Lift ADDR LOAD
4. Lift CONT; the terminal will now type:

```
ØØØØ #XXXX
     READY
```

You may now begin conversing with the BASIC interpreter.

Explanation of program

Statement 10:

The computer will type on the Teletype:

PLEASE TYPE THE PRINCIPAL AND INTEREST RATE

Statement 20:

The computer will type a "?" and wait while you type the principal and then the interest rate.

Statement 30:

(P times I) plus P will be computed.

Statement 40:

The computer will type out

AMOUNT IS (and the value for the amount)

Statement 50:

The computer will then "loop" back to statement 10 and repeat the above operations.

Statement 60: indicates the last instruction.

3.3 STATEMENT NUMBERS

For a stored program each statement must be preceded by a statement number (1 through 2047). You may type statement numbers in any order. The program will run in numerical order.

3.4 VARIABLES

Variables in CINET-BASIC are denoted by any single letter. Thus A, B, X, and N are variables. A variable may also have a subscript. The subscript is designated by immediately following identifiers with the subscript value enclosed in parentheses.

Examples:

A (3)
A (M)
X (39)
M (I)

Variations from other basics:

Only single subscripts allowed (range ±2046)
Subscript may be computed; however, only the integer value within the accepted range will be used.

Examples:

A (I+2)
X (3*4-3)

In the following, all examples are variable A(12)

A (12)
A (12.3)
A (12.7)

3.5 NUMBERS

Input

Numbers may be typed in with or without signs. They may be typed in fixed point, floating point or exponential form. However, no more than 6 significant digits may be given. Should an error be made while inputting a number, type a back arrow (←) and retype entire number.

FIXED POINT EXAMPLES:

5

-2701

FLOATING POINT EXAMPLES:

0.00025
3.2

EXPONENTIAL EXAMPLES:

15E32 means 15 X 10^{32}

3.25E-6 means 3.25X10^{-6} or .00000325

The limit on exponents is ± 615

Caution: Should you exceed the limit of the exponents by some iterative process, the exponent generated will reverse its sign.

Output

The printout of numbers will be given in floating point unless the value exceeds the range of 10^{-5} to 10^{9}. Numbers that exceed this range will automatically be printed in exponential form.

Examples:

1.5

-2035.

0.3200000E11

Caution:

Values are accurate to six significant digits only; disregard all digits beyond this range.

3.6 INSTRUCTIONS

Following each statement number is an instruction. The instruction name must be followed by a space. Correct spacing in CINET-BASIC is important. However, in general, correct spacing can be achieved if the user will use the normal conventions of spacing between words and not spacing in algebraic expressions.

Instruction	Example	Explanation
A. PRINT	30 PRINT "X=",X	outputs information
B. INPUT	10 INPUT A	allows input of values for variables
C. LET	55 LET B=4*(X+3)	defines variables
D. GO TO	140 GO TO 30	unconditional jump
E. IF	70 IF A = 5 THEN 59	conditional jump
F. FOR	85 FOR I=1 TO 30 STEP 2	automatic looping
G. NEXT	37 NEXT I	end of FOR loop
H. GOSUB	127 GOSUB 300	enter subroutine
I. RETURN	320 RETURN	return from subroutine
J. DIMENSION	12 DIM A (15)	designates upper limit of subscript
K. REMARKS	5 REM I STANDS FOR INTEREST	explanation remarks only
L. STOP	43 STOP	logical stop
M. END	2000 END	last statement in program

3.6.1 PRINT

The PRINT instruction commands the computer to output information.

If PRINT is followed by a letter or series of letters separated by commas or semicolons, the values presently assigned to these variables will be typed. If PRINT is followed by any series of characters enclosed within quotation marks, the enclosed characters will be reproduced exactly.

Examples:

```
100 PRINT A
```

If "A" were first computed to be 5, the Teletype (TTY) would type:

```
5.
50 PRINT "AMOUNT IS", A
```

The TTY would type

```
AMOUNT IS 5.
```

The PRINT instruction may also be used to evaluate one or more expressions.

Examples:

```
1000 PRINT 2-5, SQR (64)
```

2-5 would be evaluated and output as -3. Then the square root of 64 would be found and output as 8. 14 spaces are reserved for each numerical value typed. The output would appear as

```
-3            8.
```

The PRINT statement will normally give a carriage return and line feed at the completion of typing. This carriage return, line feed, may be suppressed by ending the statement with a comma or semicolon.

Examples:

```
20 PRINT "WELL!"
30 PRINT "HI",
40 PRINT "THERE"
```

The TTY will type the following, suppressing the carriage return and line feed, after the "HI":

```
WELL!
HI THERE
```

The PRINT instruction can be used to give line spacing.

Example:

```
1000 PRINT
```

The TTY would give only a carriage return and line feed.

Variation from other Basics:
There is no distinction made between a comma and a semicolon in a PRINT statement. 14 spaces are reserved for each variable regardless.

No attempt is made to give an automatic carriage return, line feed, at the end of 72 characters (width of line on TTY). The carriage return line feed is only given when the PRINT statement is complete. It is therefore necessary for the programmer to be aware of the length of his printout.
See Appendix A.

3.6.2 INPUT

The INPUT instruction should be followed by a letter or series of letters separated by commas. When the instruction is executed, the computer will type a question mark "?", stop, and wait for you to type the values you wish assigned to each variable.

Example:

```
21 INPUT A, B
```

On executing this instruction, the following would result:
The computer prints a "?" and waits for input from the keyboard. If you type a 5 terminated with a comma, the computer assigns the value 5 to the variable A, types a "?" and waits again. If you type 70.2 terminated by a comma, the machine sets B = 70.2 and types a carriage return. The result on paper looks like this:

```
?5,?70.2,
```

A value may also be terminated with a space, carriage return, or semicolon.
Variations from other BASIC's: see Appendix B.

3.6.3 LET

The LET instruction is used to define a value for a variable. This assigned value may be a single number or an algebraic expression involving some arithmetic operations. The arithmetic operations possible are:

sign	operation	example
+	addition	A + Z
-	subtraction	3 - 5.03
*	multiplication	B * C
/	division	12/3
↑	exponentiation	A ↑ 2 (means A^2) only integer portion of an exponent will be used
( )	parentheses may be used in pairs to clarify any algebraic expression.	

Order of priority of operations:

1. Values inside parentheses
2. Exponentiation
3. Multiplication
4. Division
5. Addition and subtraction

Examples:

32 LET S = 5

Defines the variable S as equal to 5.

40 LET Y = 4*A*X↑2+X

Defines Y to equal $4AX^2+X$

55 LET Y = 2*(2*A-B)/3

Defines Y to equal $\frac{2(2A-B)}{3}$

3.6.4 FUNCTIONS

In addition to the five arithmetic operations, the computer can evaluate several mathematical functions. These functions are given special 3-letter names.

Functions:	Interpretation:
SIN(X)	Find the sine of X
COS(X)	Find the cosine of X
ATN(X)	Find the arctangent of X

X in the above functions is assumed to be measured in radians. Should you wish to find the sine of an angle in degrees, the conversion factor $\pi/180$, may be used.

Example:

10 PRINT SIN(15*3.14159/180)

This will give the sine of 15 degrees

EXP(X)	Find e^X or $(2.718281)^X$
LOG(X)	Find the natural log of X
ABS(X)	Find the absolute value of X
SQR(X)	Find the square root of X
INT(X)	Find integer part of nummber in the range $\pm$2046
RND(X)	Find a nonstatistical pseudo-raudom number between $\pm$1

Caution: The last letter of the function name must be followed immediately by a left parenthesis.

Variations from other BASIC'S:

Multiplication and division are given different priorities; therefore, A/B*C gives a value for A/(B*C).
The tangent function is not available; however, it may be created by SIN(X)/COS(X).
See Appendix C.

3.6.5 GO TO

GO TO

A GO TO instruction is always followed by a statement number, directing the computer to that statement. The computer will execute instructions in numerical order unless re-directed by a GO TO statement or an IF, THEN statement.

Example:

50 GO TO 25

3.6.6 IF, THEN

This statement allows the computer to make a decision. Each statement must be of the form:

IF (variable) (relation) (expression) THEN (line number)

The relation may be made up of a >, <, =, or any combination of the three.

Examples:

20 IF X = 0. THEN 85

The computer will go to statement 85 if X = 0.; otherwise, it will execute the next statement after 20.

34 IF X<= 3+5*A THEN 97

If X is less than or equal to 3+5*A, statement 97 will be executed. If not, the next statement after 34 will be executed.

Variations from other Basics:

The left member may only be a variable.

Caution: Do not space between relation signs.
Space before and after THEN.
If zero is used in an IF statement, use it with a decimal point.

3.6.7 FOR

This instruction is used for convenience in setting up program loops and iterations. It must be used in conjunction with a NEXT statement. The general format is:

100 FOR A = X TO Y STEP Z

The variable A is initialized to the value of X, then those statements following the FOR are executed until a NEXT statement is encountered. When a NEXT statement is found, control is sent back to the preceding FOR. The Value A is then incremented by Z and compared to the value Y. If A is less than or equal to Y, then those statements following the FOR will once more be executed. This process will be repeated until A is greater than Y. At that time the statements following the NEXT will be executed. X, Y, and Z may be algebraic expressions. Z must have a positive value. If STEP is omitted, then the increment will be one.

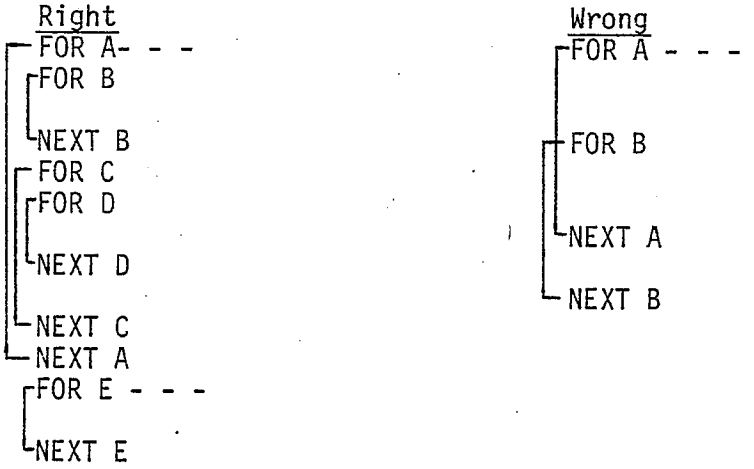
Example:

```
10 FOR A = 1 TO 2↑2 STEP 2.5
20 PRINT A
30 NEXT A
```

When RUN is typed, this program would give:

```
1.
3.5
```

FOR, NEXT statements may be nested as diagrammed on the left.



When using a FOR loop, the loop may be terminated prematurely with an IF statement. However, when exiting from nested FOR statements prematurely, the entire nest should be terminated.

Caution: Space before and after "TO" and "STEP".

3.6.8 NEXT

Used only with FOR statements. The general form is:

```
20 NEXT A
```

(See FOR)

3.6.9 GOSUB

When a particular part of a program is to be performed more than one time, or possibly at several different places in the program, it is most efficiently programmed as a subroutine. The subroutine is entered with a GOSUB statement, where the number is the first line number in the subroutine. The subroutine must logically end with a RETURN instruction. This transfers control back to the statement following GOSUB.

Example:

```
10 PRINT "A",
20 GOSUB 100
30 PRINT "B",
40 GOSUB 100
50 PRINT "C"
60 STOP
100 PRINT "XXX"
110 RETURN
```

This program would give:

```
AXXX
BXXX
C

READY
```

3.6.10 RETURN

The RETURN must be the last statement in a subroutine.

Example:

```
300 RETURN
```

See GOSUB

3.6.11 DIMENSION (DIM)

DIM is used to reserve space for subscripted variables greater than 10.

Example:

```
20 DIM A(20), B(15)
```

Variations from other BASIC'S: See Appendix J.

3.6.12 REMARKS (REM)

This statement is used to allow the programmer to insert explanatory remarks in a program. The computer will completely ignore the line when RUN is typed.

Example:

```
100 REM STATEMENT 110 MAY BE CHANGED TO GIVE NEGATIVE ROOTS.
```

3.6.13 STOP

The STOP instruction is used to designate a logical stop in the middle of a program.

Example:

```
30 STOP
```

3.6.14 END

The END instruction is used to designate the last statement in a program.

Example:

```
99 END
```

Variations from other BASIC'S: See Appendix M.

SECTION 4.0

DIRECT MODE

If you wish to run a program that is only one step long such as:

PRINT SQR(49)

you may use direct mode in CINET-BASIC. This is accomplished by simply typing the above instruction (without a statement number) followed by a carriage return. In this example, "7." will be printed immediately. The instruction is executed but not stored for future use.

SECTION 5.0

COMMANDS

There are certain commands that are intended for direct mode only. These are:

Commands	Example	Explanation
A. RUN	RUN cr	run stored program
B. LIST	LIST cr	LIST out stored program
C. EDIT	EDIT 20 cr	Set up statement 20 for EDIT
D. ERASE	ERASE ALL cr	ERASE entire stored program
E. INTERRUPT	(hold CTRL key and depress C key)	This interrupts CINET-BASIC and returns control to user

5.1 RUN

RUN will direct the computer to execute the stored program.

Examples:

RUN This executes a stored program starting with the lowest-numbered instruction.

RUN 50 This executes a stored program starting with statement 50 and ignoring all lower-numbered instructions.

5.2 LIST

The LIST command will direct the computer to list out the stored program. LIST followed by a statement number would direct the computer to list from that statement number on.

Examples:

LIST	Lists the complete stored program
LIST 37	All instructions with statement numbers equal to or greater than 37 will be listed.

5.3 SAVING PROGRAMS:

To save on tape a CINET-BASIC program that is stored in the computer, the user should:

1. Type LIST (DO NOT depress the RETURN key).
2. Turn on tape punch.
3. Type several @ signs to get leader tape (press the Shift, REPEAT, and P keys in that order; release in reverse order).
4. Depress the RETURN key.

When user's program has been typed and punched out:

5. Type several more @ signs to get trailer tape.
6. Turn off tape punch.

5.4 EDIT

This command must be followed by a statement number. The statement designated will be set up for edit upon receiving a carriage return. The user then types a search character. The computer will type out the contents of that line until the search character is typed. The user may then perform any of the following optional operations:

1. Type in new characters. They will be added to the line at the point of insertion.
2. Type a FORM (CTRL/L). This directs the computer to proceed to the next occurance of the search character.
3. Type a BELL (CTRL/G). After this the user may change the search character.
4. Type a RUBOUT. This deletes characters to the left. One character is deleted for each time that the user strikes the RUBOUT key.
5. Type ← (SHIFT O). This deletes all the line to the left margin (not the line number).
6. Type RETURN. This terminates the line, deleting characters to the right margin.
7. Type LINE FEED. This saves the remainder of the line from the point LINE FEED is typed.

A completely new statement will replace an existing one by typing the old statement number followed by the new statement.

5.5 ERASE

Erase must be followed by some parameter.

Examples:

ERASE 30

The above command will erase from the stored program statement number 30.

ERASE ALL

The above command will erase the entire stored program.

Note: Typing a statement number alone followed by a carriage return will delete from storage the contents of that line. It will, however, leave the statement number in the stored program. This line number is ignored when RUN is typed.

5.6 INTERRUPT

You may interrupt CINET-BASIC at any time to return control to the user by typing a CTRL/C.

SECTION 6.0

ERROR AND CONTROL MESSAGES

These messages are given as a four-digit number followed in turn by a "#" and an additional four-digit number. The last number where applicable designates the statement number where the error occured. Number 2080 designates a direct mode error.

Example:

0206 #0020

This implies an illegal command in statement 20.

Below is a list of messages.

0000 #X	Manual start or interrupt
0031 #X	Function not available
0109 #2080	Line number too large
0167 #X	Storage filled by statement X
0195 #X	GO TO Y, and there is no Y
0206 #X	Illegal command in statement X, or NEXT without FOR
0263 #X	IF without THEN
0275 #X	Left side of equal sign is illegal in statement X
0286 #X	Too many right parentheses in LET statement
0353 #2080	EDIT X and there is no X
0453 #X	Illegal variables or illegal spacing in FOR, or IF statement
0464 #X	Operator missing in statement X or space in Subscripted Variable
0506 #X	Variable illegal; illegal spacing in FOR, IF or PRINT statements
0507 #X	Operator error in statement X
0534 #X	Parentheses error or parentheses not immediately following function
0581 #2080	ERASE with no parameter
0598 #2080	Line number too large
0716 #2080	Input buffer error
0811 #2080	Line too long or program too long
1301 #X	Attempt to take LOG of zero
1486 #X	Constant or input with too many significant figures
1641 #X	STEP used incorrectly
1642 #X	THEN used incorrectly
1657 #X	TO used incorrectly
1784 #X	Argument of function or subscript exceeds acceptable range
1832 #X	Attempt to divide by zero
1943 #X	Attempt to take square root of negative number

SECTION 7.0

PREPARING OFF-LINE PROGRAMS

If someone else is using the computer, you may punch a program tape by typing your program on an off-line Teletype with the TAPE PUNCH ON. This is accomplished by first typing several @ signs for leader tape. Then type your program. Finally type several more @ signs for trailer tape. Should you make an error while typing, a "back-arrow" (←) will erase the present line when the tape is read in the computer. Or, you may have the computer delete one or more characters from the tape by following bad characters with RUBOUT's, one RUBOUT for each bad character. (Caution: DO NOT BACKSPACE TAPE.) Now take your program tape to the computer's Teletype and insert the tape in the tape reader. Type "ERASE ALL", a Carriage Return, and place the tape reader switch in START position. The program will be quickly loaded into memory. After the tape has loaded, place the tape reader switch in STOP position.

When loading a long program (tape), it is possible to receive the error message 0716 #2080. This may be corrected by simply stopping the tape reader periodically to allow the Input Buffer to clear itself.

APPENDIX

CINET-BASIC Extensions

A. PRINT

1. It is possible to get a carriage return without a line feed. This is accomplished by placing a # (number sign) in the print statement. It must be separated from other output by commas, but not in quotes. This will cause the TTY to give a carriage return only.

Example:

```
10  PRINT "IS    RIGHT?",#
20  PRINT "  ",A
```

If A is 5, then the TTY will type

IS 5. RIGHT?

This is very useful in plotting or graphing.

2. A carriage return may be substituted for a " sign if the quote is the last character in a line.

B. INPUT

The control characters " and # of the PRINT statement are also operative on the INPUT statement.

Example:

```
150 INPUT "TYPE THE AMOUNT OF INCOME", A
```

The TTY would type the following and then wait for you to assign a value to 'A' as shown below.

TYPE THE AMOUNT OF INCOME?

C. LET

It is possible in CINET-BASIC to allow one LET statement define a series of variables by separating each equation with a comma or semicolon.

Example:

```
10 LET A = 2, B = 3, C = B*A
20 PRINT A, B, C
```

This program would give

2. 3. 6.

APPENDIX

J. DIMENSION (DIM)

The DIM statement may be omitted, as it is not necessary to reserve space in CINET-BASIC

M. END

In CINET-BASIC the END instruction is not nesessary. However, the computer will not acknowledge the completion of the program with READY, unless END or STOP is used.