of your program. You'll know at once what simple variables and strings you've used and what values you've assigned to each.

```
FIND A$,100-200
  110 A$="HELD"+B$
  180 B$=MID(A$,I,12)
  200 INPUT A$

READY.
```

FIND Like **LIST**, this command will show a set of lines. But the FIND command is followed by specifying a character string. Those lines, and only those lines, containing a desired character string will be listed on you PET's screen. If you were to type **FIND A$, 100-500** your PET's screen would display all lines between line numbers 100 and 500 that contain **A$**.

```
APPEND "INPUT"

PRESS PLAY ON TAPE #1
OK

SEARCHING FOR INPUT
FOUND INPUT
APPENDING

READY.
```

APPEND You've already worked hard to develop a number of programs and, of course, you've saved them on tape. You're working on a program that's now in memory. Just type in **APPEND** "*program name*" and all statements in that program will now follow the program in memory. No need to retype; no opportunity for errors. Save any set of statements or subroutines onto a tape, using the normal **SAVE** command; then recall them with the **APPEND** command and add them permanently to your program.

From an Expert:

"...my **Toolkit** *arrived...the ROM was thoughtfully placed in a conductive plastic pin carrier, protecting it from the possibility of bent pins and electrostatic damage.*

Also enclosed was a 34 page book of documentation. Professionally printed with a firm slick cover, it was a delight compared to some of the documentation that we have all seen in the mass micro market. I immediately got the impression that this company cared about what their customers thought about them. At this point I was favorably impressed and developed some confidence in the product and the company...

In my opinion the investment for your PET/CBM is more than worth it. The product performs great and with the guarantee...you can hardly go wrong."

...An author and frequent contributor to computing magazines

The Toolkit consists of two kilobytes of ROM firmware on a single chip.

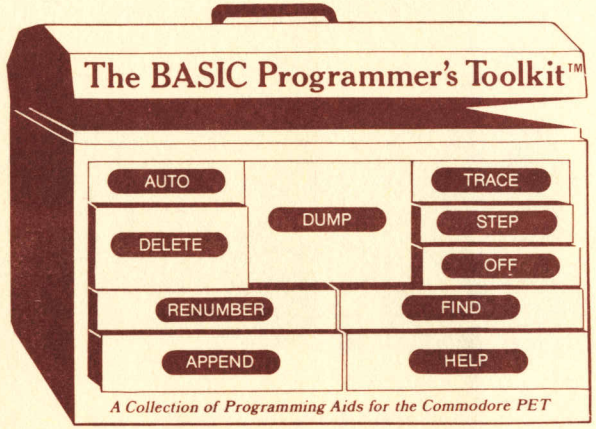
For the 16K and 32K PETs, the ROM chip plugs into a spare socket on the main circuit board inside your PET.

For the 8K PET, **the Toolkit** is mounted on a special printed circuit board with edge connector and attaches to the memory expansion port on the right side of the PET, and to the second cassette interface.

Memory addresses in ROM: $B000 to $B7FF; addresses in the PET RAM: $03E0 to $03FF

AVAILABLE AT:

Increase Your PET's IQ



with the

BASIC Programmer's ToolKit™

The Toolkit is a collection of machine language firmware aids designed to enhance the writing, debugging and polishing of BASIC programs for the PET. **The Toolkit** offers additional ROM storage, avoiding any need to load tapes or to give up valuable RAM storage.

The Toolkit *is fully assembled. It is not a kit and requires no special tools to install.*

The Tools in the BASIC Programmer's Toolkit™

Power up your PET, execute the command **SYS** with the appropriate address and your *new* PET is off and running with these commands:

```
AUTO 100,25
   100 FOR I = 1 TO 10
   125 GOSUB 300
   150 ■
```

AUTO This command is followed by a series of optional parameters specifying where you want to enter lines and how far apart you want them. Your PET will automatically respond with a line number prompt. You won't have to enter the line numbers; you won't have to worry about errors of screen editing.

```
DELETE 200-350

READY.
LIST 200-350

READY.
```

DELETE Like **LIST**, this command is followed by a range of line numbers . . . and every one of the lines within the range of numbers will be removed instantly from your program. No longer will you have to type a line number, press **RETURN**, type the next line number, **RETURN**, next line. . . .

```
RUN

?DIVISION BY ZERO ERROR IN 500
READY.
HELP
 500 J = SQR(A*B/C)

READY
```

HELP How many times have you wanted to scream "HELP!" when your PET couldn't interpret your program and all it would say was: **?SYNTAX ERROR**? Fret no longer: Now just type in **HELP**. The line on which the error occurs will be shown and the erroneous portion of the line will be indicated in reverse video on the screen. Truly a great help in any learning or school situation.

```
LIST

  10 GOSUB 99
  15 PRINT I
  16 GOTO 10
  99 INPUT J
 100 IF J = 0 THEN END
 200 I = SQR(J):RETURN
READY

RENUMBER 100,10

READY.
LIST

 100 GOSUB 130
 110 PRINT I
 120 GOTO 100
 130 INPUT J
 140 IF J = 0 THEN END
 150 I = SQR(J):RETURN
READY.
```

RENUMBER Now you can change all line numbers— and all references to those numbers—instantly, as, for instance by evenly spaced increments of 100 or 25 or 10. . . .

```
TRACE                    #100
                         #110
READY.                   #150
RUN                      #160
                         #175
ENTER YOUR NAME? JIM     #200

HI JIM.

HOW OLD ARE YOU?
```

TRACE Now you can see precisely the order and sequence in which your program is being executed. You can also stop the program at any point and record the sequence. Type in this command and your PET will keep a record of the line numbers of the last six statements executed. These last six statement numbers will appear in a small rectangular window in the upper right hand corner of your screen.

STEP Again your line numbers are displayed in the upper right hand corner of the screen in this version of **TRACE**. But now your PET executes just one statement and pauses until you press **SHIFT**. Then it proceeds to the next statement.

OFF This command will stop either **TRACE** or **STEP**.

```
RUN

READY.
DUMP
A1 =  10
BW = -6.1
C$ = "HI"

READY.
```

DUMP During or after running a program, this command will display the names and values of all variables used in the execution