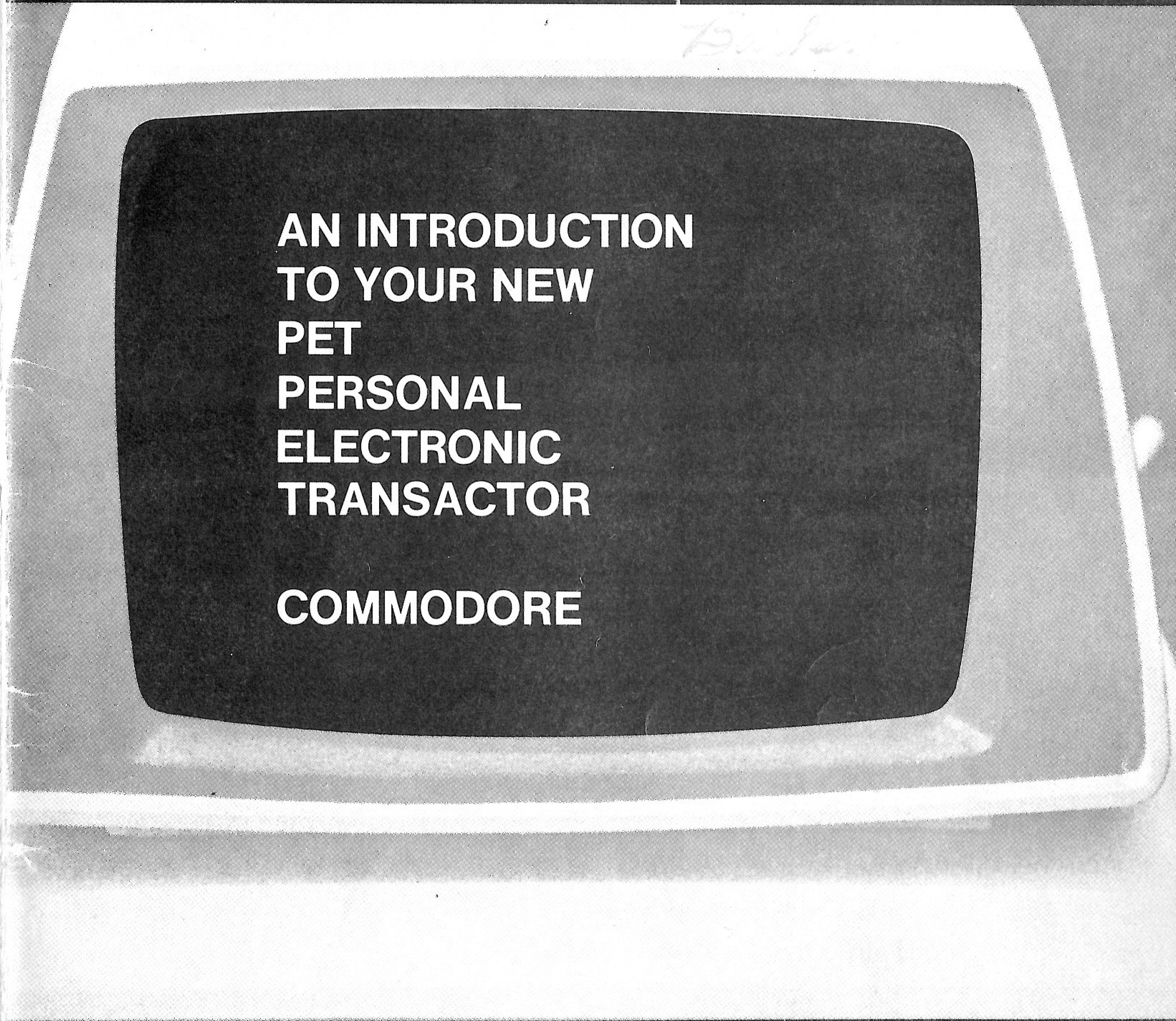
Butler



AN INTRODUCTION TO YOUR NEW PET PERSONAL ELECTRONIC TRANSACTOR

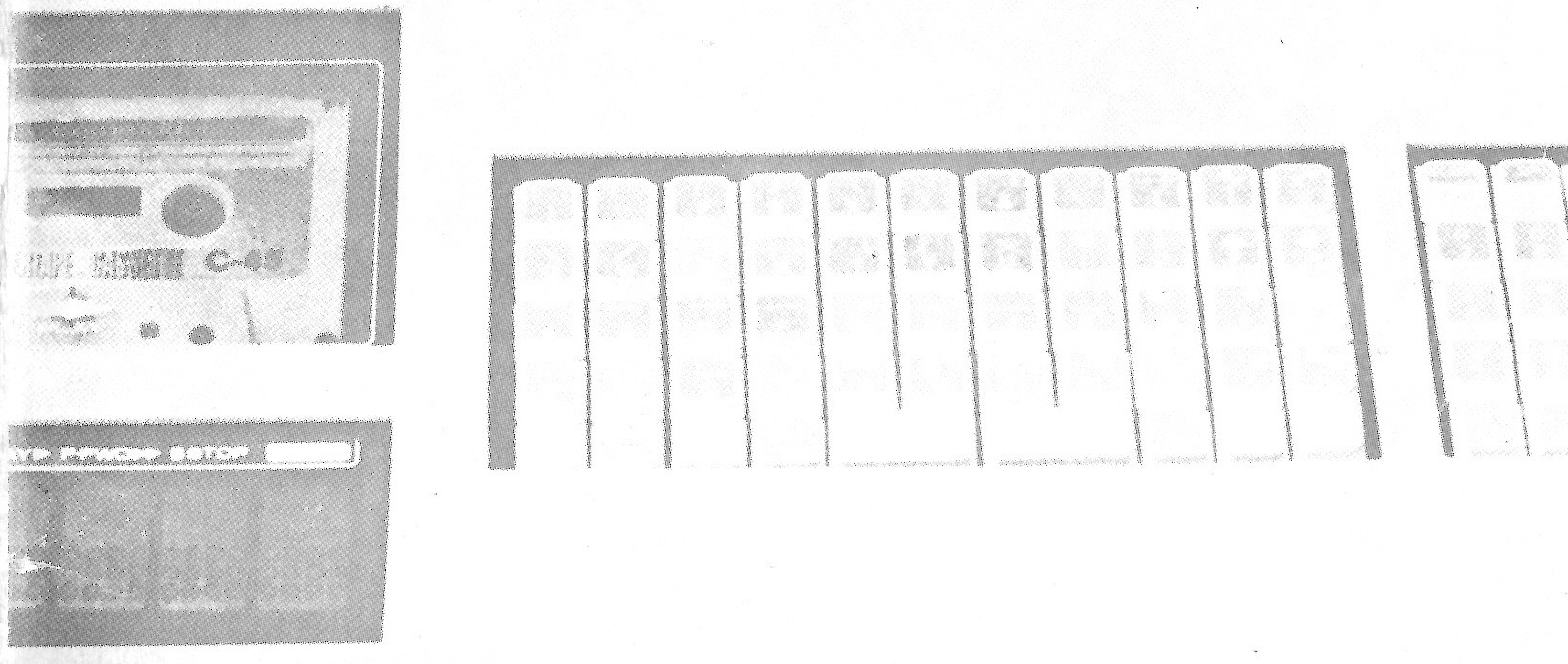
COMMODORE

commodore

PET

2001 Series

personal computer



The potentials of your PET are virtually limitless. This booklet, by its very nature, is limited. Many questions will arise that this booklet has not covered or even anticipated.

Write to us at Commodore with your questions. We will answer many that you and other users will pose with a newsletter we'll be sending out from time to time to users. Or your answer may be incorporated in a booklet for which we'll charge a nominal fee . . . but we will call this to your attention.

User Program
Commodore Systems
901 California Avenue
Palo Alto, California 94304

Table of Contents

APPENDIX

Congratulations . . .

We congratulate you on your purchase of a Commodore PET computer and welcome you to the growing legion of PET users.

The number of applications for the PET is as great as your imagination and the number of programs available.

We shall attempt, in this modest booklet, to introduce you to the art of doing your own programming and to the PET method of using prepared programs.

We do recommend you to the brief bibliography in this booklet's appendix. Enjoy learning about your PET and the world of computers as much as this booklet's authors did.

By the time you're through with this booklet you should be on your way to enjoying and using your PET to its fullest.

Unpacking your PET and turning it on

Please check the carton for any special unpacking instructions. And examine carefully your PET for any concealed damage. If there is any, report this IMMEDIATELY to both Commodore (or your dealer) and to the shipping agent.

Remove your PET from its protective shipping carton, and place it on the counter, desk or other suitable surface, then plug it into any standard, grounded electrical outlet.

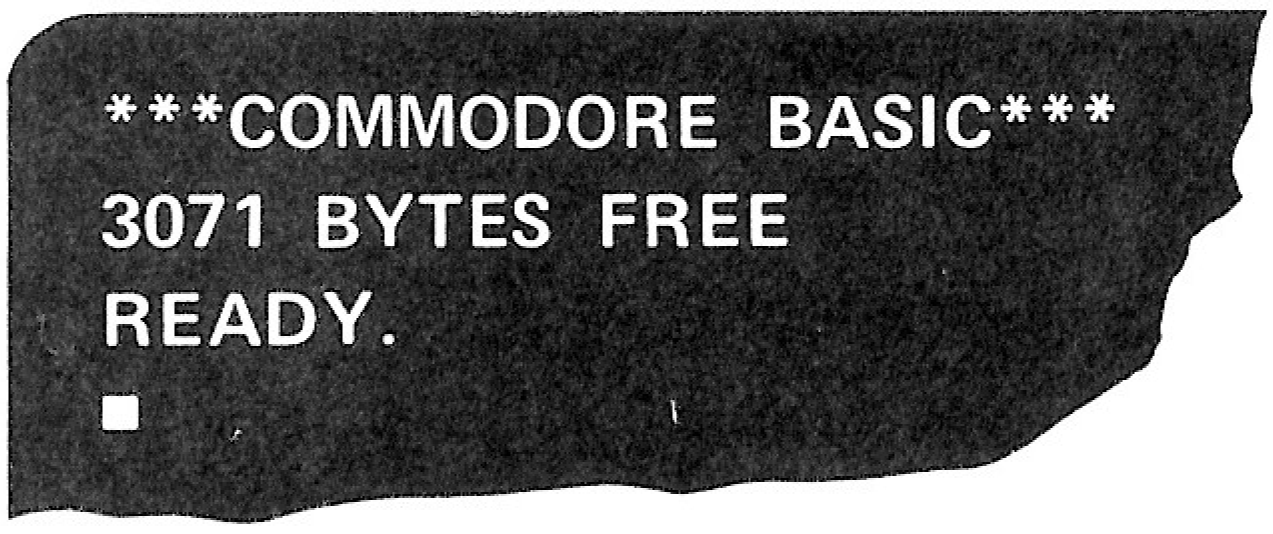
Push the rocker switch, in the lower left rear where the line cord enters the unit, to the power-on position.

> *(There's a white dot on the power switch to indicate it's in the power-on position.)*

Momentarily you may be able to see that the PET television display contains a collection of random letters and symbols. This is normal. On "Power up," PET's electrical circuits have to wake up slowly before they can function and clear the screen.

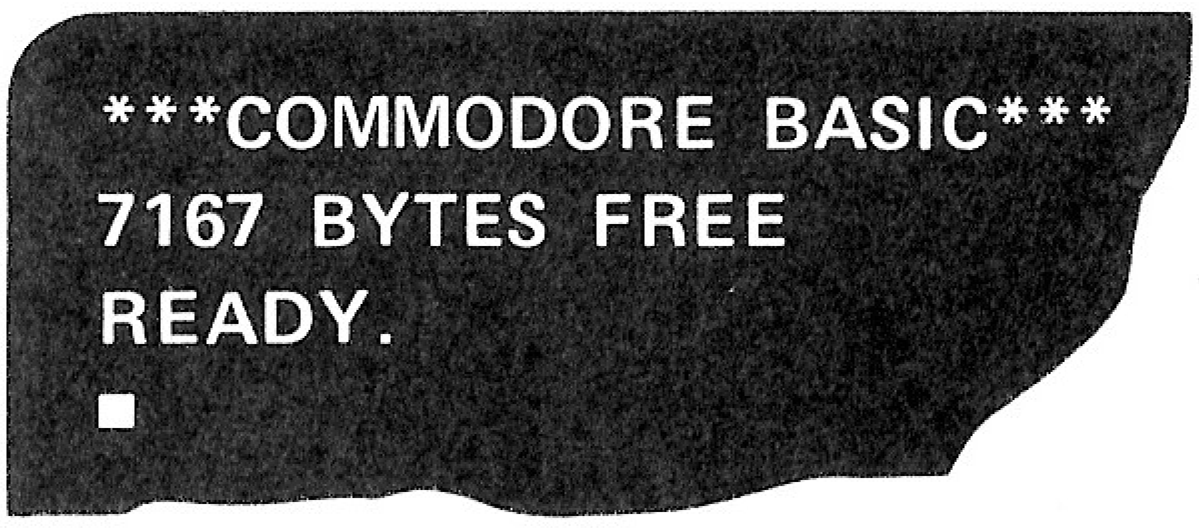
Almost as fast as you can blink an eye, the screen will clear and one of the two messages below will be printed out in white letters on the black screen. The second line of the message will vary depending on which memory option of PET you are using. It tells you how much memory space is free for you to use.

4k power-on display



```
***COMMODORE BASIC***
3071 BYTES FREE
READY.
■
```

8k power-on display



```
***COMMODORE BASIC***
7167 BYTES FREE
READY.
■
```

(A byte is the fundamental data element of the PET computer and corresponds roughly to one letter or digit of information. For the curious: the 4k model should show – in theory – "4096 bytes" and the 8k, "8192 bytes." But a few hundred are used by the PET internally. The balance shown "3071" and "7167" are net available bytes.)

If you fail to get the power-up display the first time, try turning the power switch *slowly* off then back on. Rarely will PET fail to respond to this, but if it does, turn to the *Hints if You Have a Problem* section at the back of this pamphlet.

Most noticeable in the display is a flashing white square called a **cursor**. Whenever PET is waiting for some keyboard information, the cursor will begin blinking and this is where the next character will appear if it is typed in.

Touring the keyboard

But, before you can speak to your PET, we need to take a brief tour of the keyboard.

Each key has a thin, transparent plastic film covering the keytop which should be removed. This protection was left in place to protect the keys against scratches during shipping. To remove the film, carefully peel it off by using the sticky side of a piece of masking tape so as to avoid scratching the keytops.

Keytop legends bear much resemblance to those of a standard typewriter keyboard, but there are a few differences.

The letters are all in virtually the same place as on a standard typewriter keyboard, but, for your convenience in numerical computations, the numbers are separate and laid out very much like a calculator keyboard. (See Figure 1.)

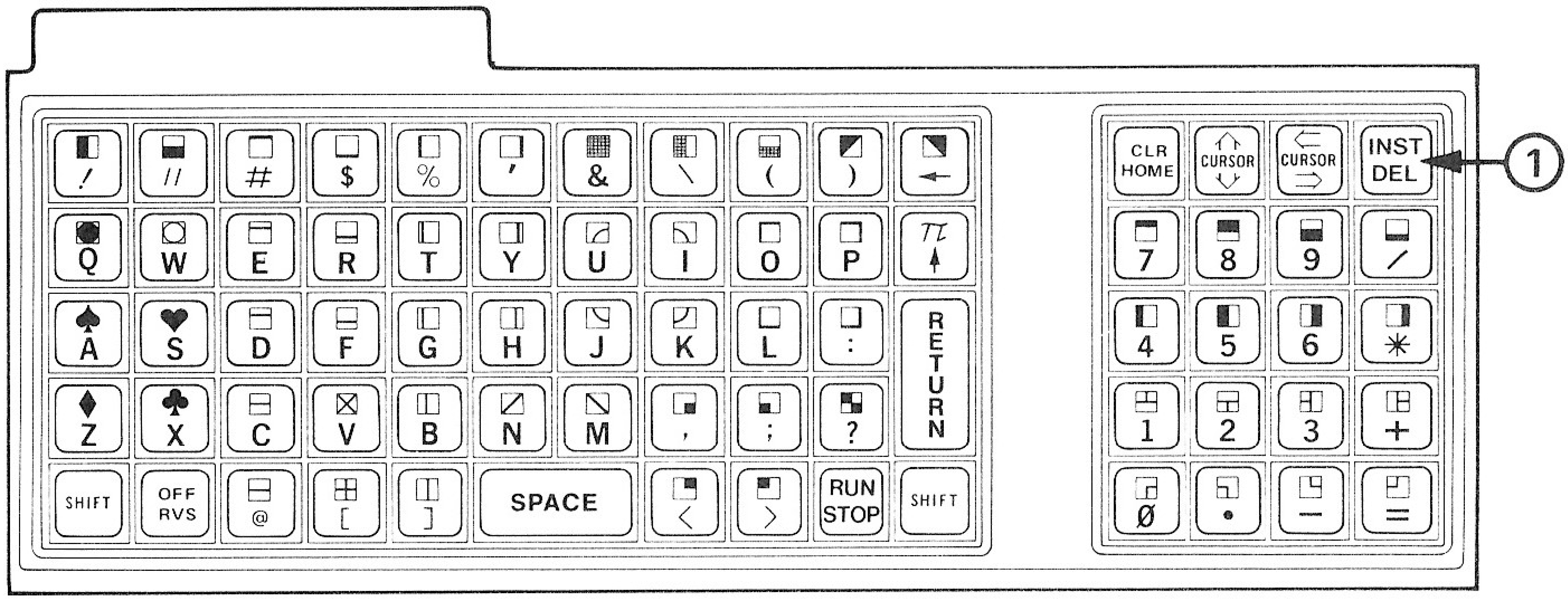

! " # $ % ' & \ ( ) ←
Q W E R T Y U I O P ↑
A S D F G H J K L : RETURN
Z X C V B N M , ; ?
SHIFT OFF RVS @ [ ] SPACE < > RUN STOP SHIFT
CLR HOME CURSOR CURSOR INST DEL
7 8 9 /
4 5 6 *
1 2 3 +
Ø • — =
1

Figure 1. The keyboard

On most typewriters, if you strike a letter key, without shifting, you will get a lower case letter. On your PET, if you press a letter key without shifting, you will get the capital letter. (See Figure 2.)

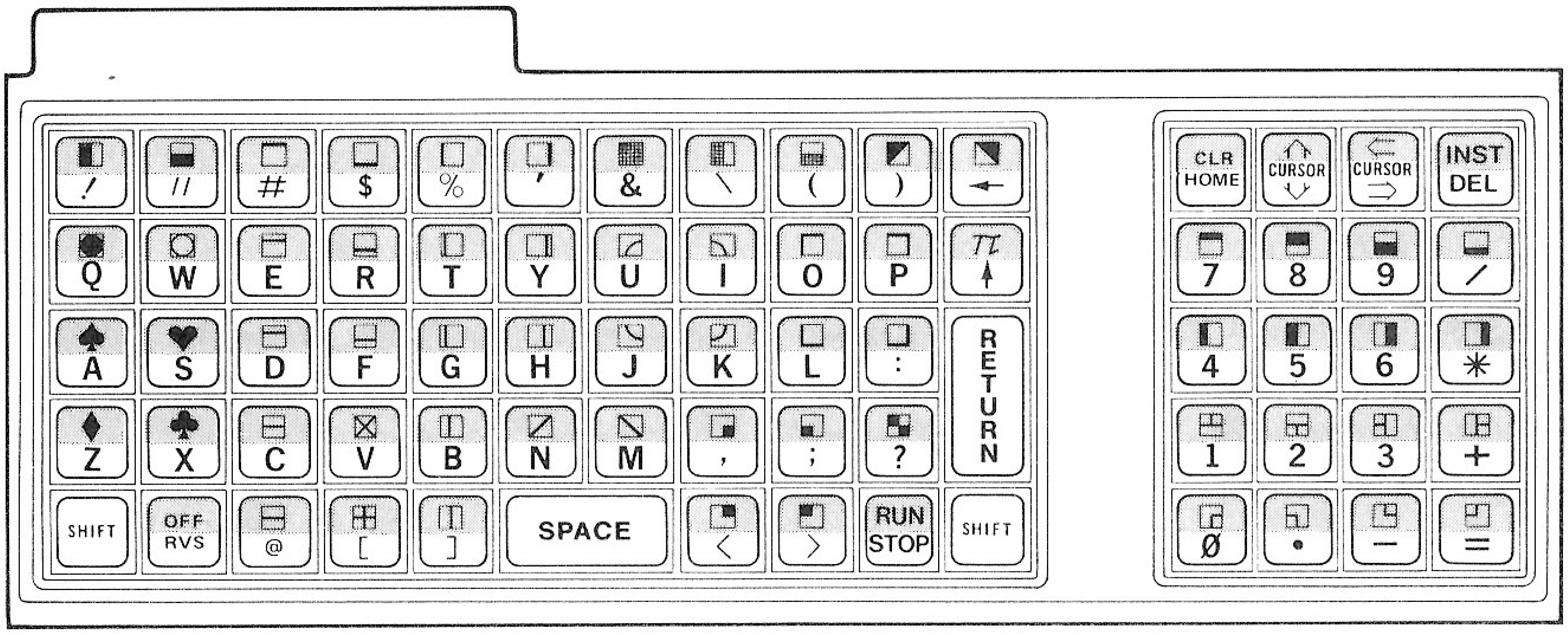

! " # $ % ' & \ ( ) ←
Q W E R T Y U I O P ↑
A S D F G H J K L : RETURN
Z X C V B N M , ; ?
SHIFT OFF RVS @ [ ] SPACE < > RUN STOP SHIFT
CLR HOME CURSOR CURSOR INST DEL
7 8 9 /
4 5 6 *
1 2 3 +
Ø • — =

Figure 2. The characters without shifting

If, on the other hand, you simultaneously press the shift key and a letter key, you will get the particular graphic that appears above the letter: (See Figure 3.)

The graphics characters are a special set of symbols unique to PET. They are used to draw pictures and lines on the screen and to perform simple animation. The graphics can be printed on the screen just like any other letter or digit.

For now, locate the A key and press it a number of times to get a row of characters – AAAAA – on the screen. (Do *not* use the SHIFT key. If you did, you'd get ♠.)

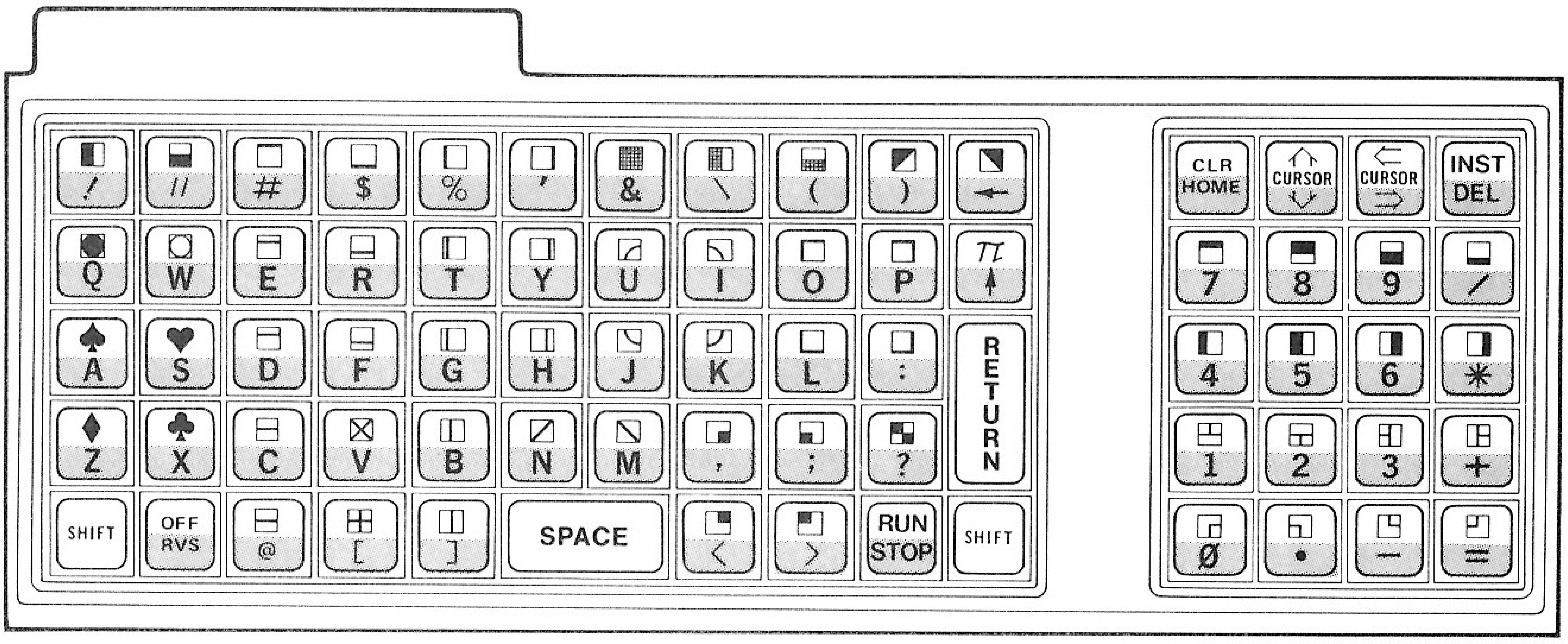

! " # $ % ' & \ ( ) ←
Q W E R T Y U I O P π ↑
A S D F G H J K L : RETURN
Z X C V B N M , ; ?
SHIFT OFF RVS @ [ ] SPACE < > RUN STOP SHIFT
CLR HOME CURSOR CURSOR INST DEL
7 8 9 /
4 5 6 *
1 2 3 +
Ø • — =

Figure 3. The characters with the SHIFT key in use

Next, press the INST DEL key labeled ① on the keyboard illustration (Figure 1). Type a different letter. Then press INST DEL again. Did you see the character erase?

(Note again: without shifting, you're getting DELETE. Shifting would get you INSERT.)

Play a little game where you type in more letters and DELETE them too.

Remember that no matter what key you press, *there is no way to damage the insides of your PET by normal keyboard operation.* (Of course, PET is not intended to survive hard falls or attacks with sharp objects – but with normal care it will give you years of service.) Do not ever be afraid to experiment.

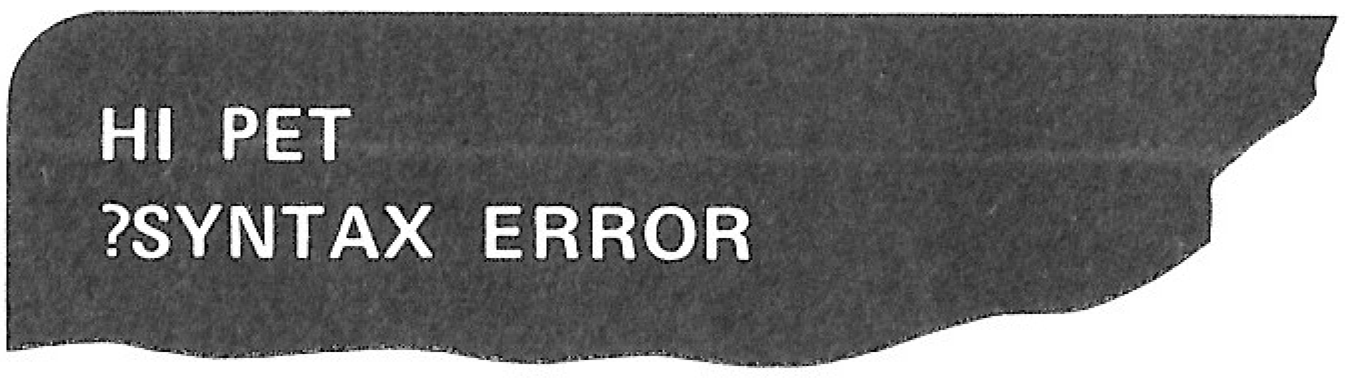
Test out the keyboard by trying the following sequence of keystrokes. Don't worry about making typing mistakes; you already know how to correct them.

Exercise 1 — Testing the keyboard

H I SPACE P E T RETURN

The RETURN key is a special signal to PET that you have finished typing a line and it should do something with it. This feature allows you to edit the line and get it correctly typed before your PET can act on it.

The important thing about this exercise is to get the following display on the screen after you've done:



```
HI PET
?SYNTAX ERROR
```

Try it again if you wish. PET is just telling you that it does not understand what you said. PET speaks a language called BASIC which was invented by some people at Dartmouth University especially for making the resources of a computer quickly and easily available to those with no previous experience. "Syntax" is, of course, the same word you encountered when studying grammar; it refers to the rules of language. So, "Syntax Error" means you haven't followed the rules exactly. And, in BASIC, *you must be exact.*

Exploring BASIC

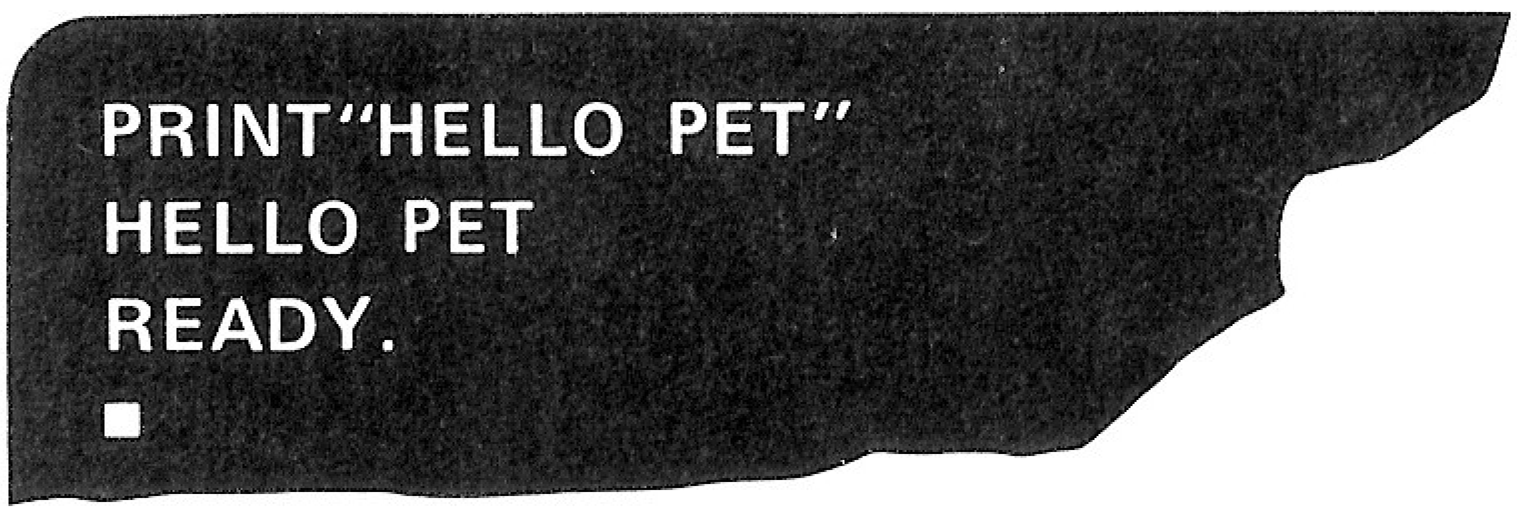
The first BASIC command we shall explore will tell your PET to PRINT something on the screen. This is one of the more useful commands, for with it you can make your PET display data, draw pictures, or play graphic games.

Now enter the following by pressing this sequence of keys. *(We'll call it "typing" from this point on.)*

Exercise 2 — Printing on the screen

PRINT"HELLO SPACE PET" RETURN

Did your PET print HELLO PET on the screen? If it did not, then try it again. PRINT is a command which tells your PET what to do with the rest of the line. This example has a message between quotes. The quotes tell your PET to print out the message exactly as it appears within the quotes without any further processing.



```
PRINT"HELLO PET"
HELLO PET
READY.
■
```

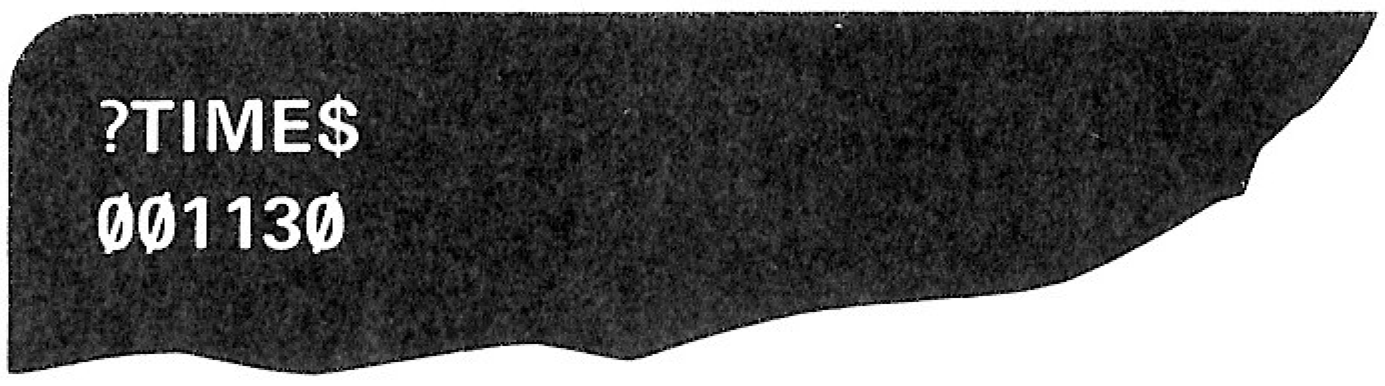
Exercise 3 — Using the built-in clock

Now, let us speak to your PET in BASIC and get it to tell you what time it is. Your PET has a built-in clock that starts from 0 the moment you turn on the computer.

To discover the elapsed time, type:

?TIME$ RETURN or PRINTTIME$ RETURN

The [?] is a shorthand which you may use instead of always typing PRINT when you want your PET to print something. The [$] at the end of the word TIME tells PET to print the time in hours/minutes/seconds. Though the elapsed time may be different, you should see a display something like this:



```
?TIME$
ØØ113Ø
```

The first two digits are elapsed hours, the second two digits are elapsed minutes, and the last two digits are elapsed seconds – which, in the above example, means that PET has been running for 00 hours, 11 minutes, and 30 seconds. The time you see on your PET, however, will depend entirely on how long you have had it on thus far. PET's clock is crystal-controlled and very accurate. It is also a 24 hour clock which means it will count up to 23:59:59 then roll over to 00:00:00.

Exercise 4 – Setting the clock

It is very easy to set your PET clock. Assume it will be 12:30 p.m. in a few seconds. Press the following sequence of keys:

```
T I M E $ = " 1 2 3 0 0 0
```

When the designated time (12:30 p.m.) comes up on your watch, press [RETURN] and PET will set the time.

Substitute your current local time and try setting the clock as in the previous example.

Now, whenever you type:

```
? T I M E $ [RETURN]
```

your PET will tell you the correct time. Remember that if you turn the power off, the clock will stop running and you will have to reset it when you turn the power on again. Once you have reset it, though, you have a highly accurate built-in clock available at all times. Just type in ? T I M E $ [RETURN] and there it is.

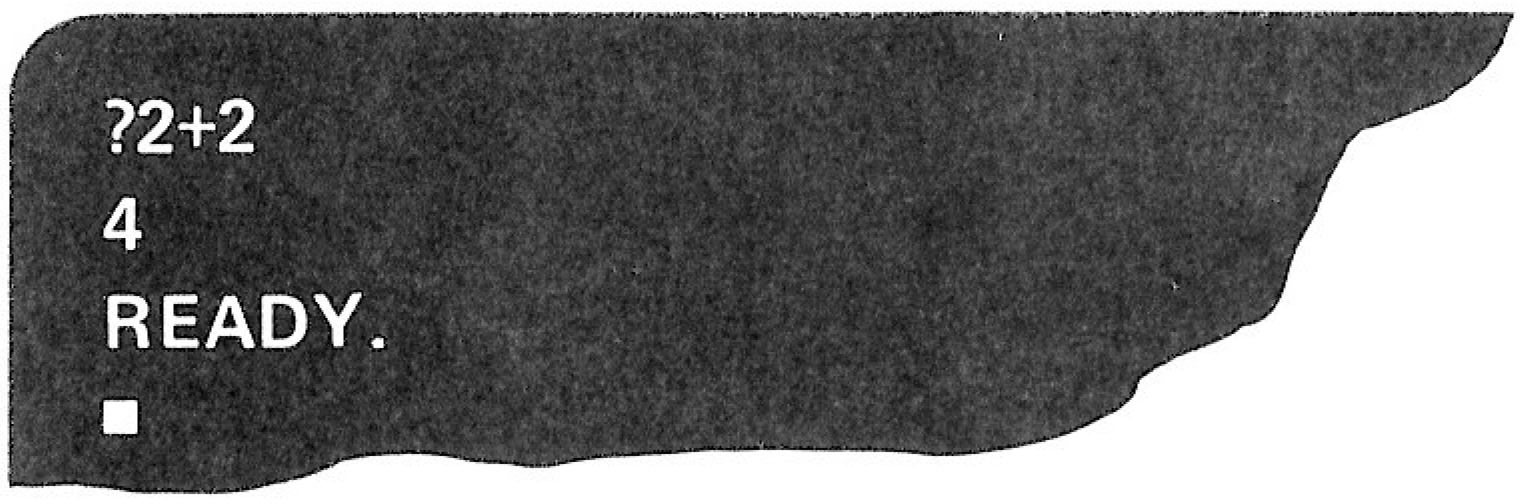
Exercise 5 – Solving mathematical problems

BASIC is essentially an algebraic language which means that you can use your PET much as you would a pocket calculator. Though it packs the power of several programmable calculators put together, it is as easy to use as a simple four-function calculator. Furthermore, everything you type into it is instantly displayed on the screen, and that makes it easier to keep track of what you are doing.

To perform arithmetic on your PET, simply tell it in BASIC to print your answer. Note how much it looks like a direct question:

```
? 2 + 2 RETURN
```

When you press the RETURN key, PET prints the result on the screen.



```
?2+2
4
READY.
■
```

Let us try another problem. But this time we'll do one which uses multiplication. An asterisk is PET's symbol for this operation:

```
? 2 + 2 * 5
```

What is the answer? Is it 20 [(2+2) x 5] or 12 [2+(2x5)]? Try it and see what happens. If the addition is done first the answer is 20, but if the multiplication is done first the answer is 12. PET could get confused, too, if it did not have a rule which it always uses: Multiplication and division (* and /) are *always* done first and addition and subtraction (+ and –) are done last. Thus, in the above example, PET does 2*5 first to get an intermediary result of 10 to which it adds 2 to get the correct answer which is 12.

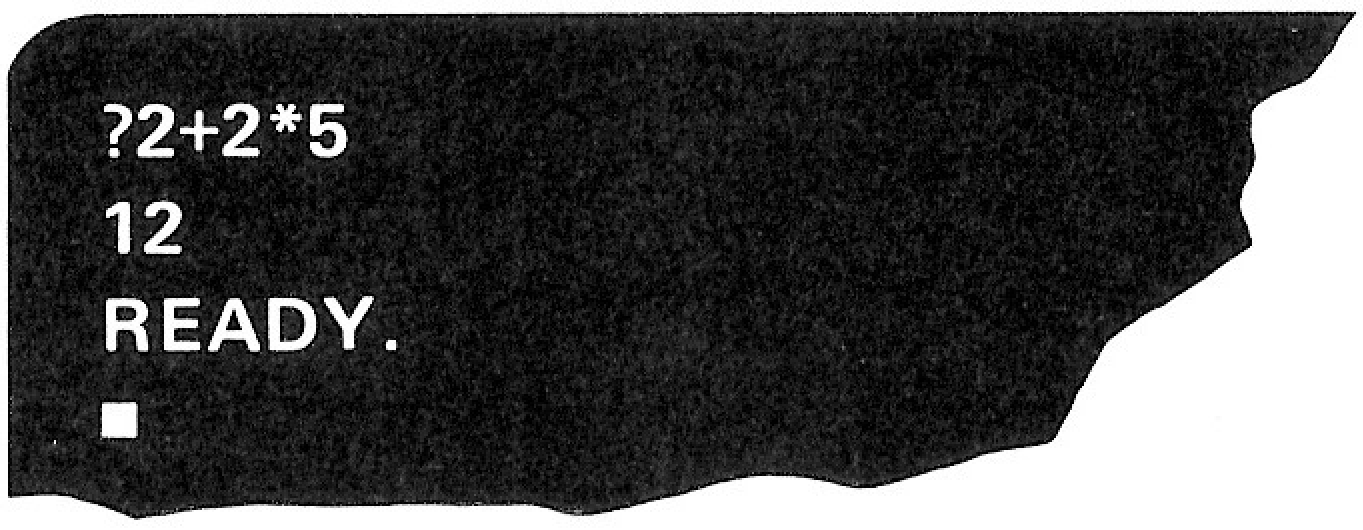
Mathematical operations are called operators. There are four operators: + *(plus),* – *(minus),* * *(times), and* / *(divided by). Notice that the last two operators are different from the ones we learned in school.* * *is used instead of* x *to avoid confusion where* x *means something else to the computer.* / *is used instead of* ÷ *because most typewriter keyboards do not have this symbol, and BASIC was first designed to be used with a keyboard that didn't have* ÷ *. Be sure you understand the order in which basic will perform the operations.*

The statement we used above:

```
2 + 2 * 5
```

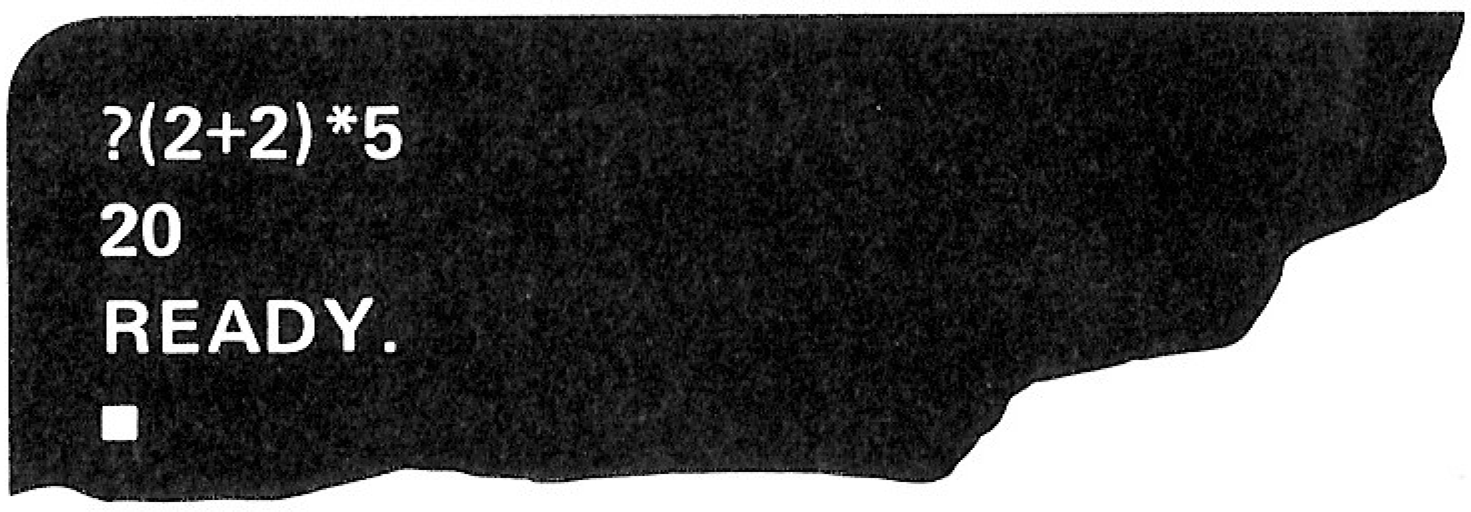
is a ***formula.*** *It describes an arithmetic procedure. The procedure is to multiply 2 times 5 and add 2. When such a for-*

mula becomes part of a computer program we call it an ***algorithm.***



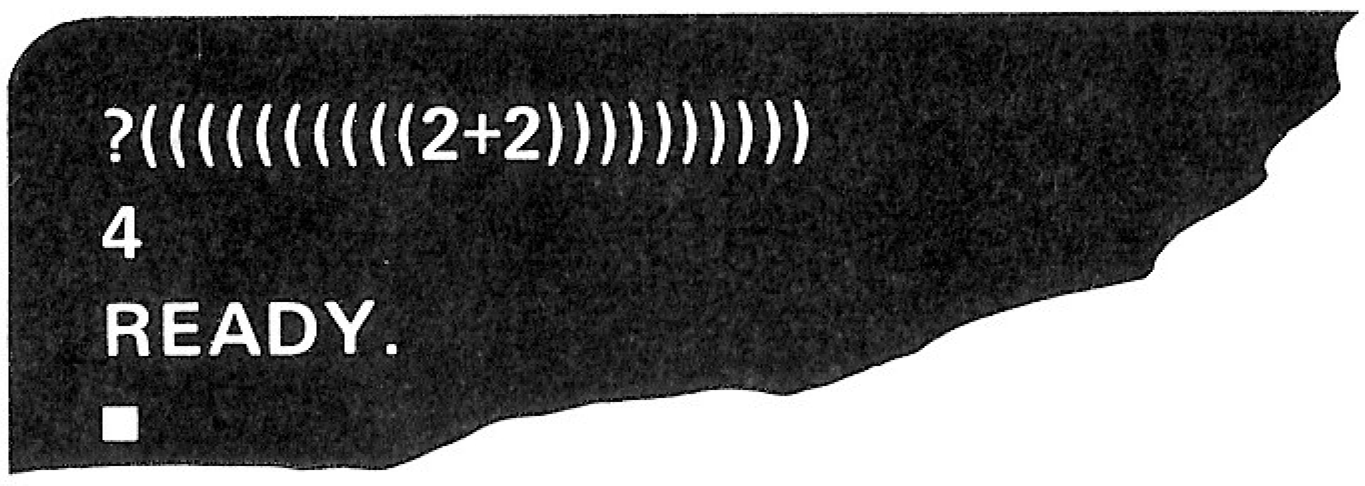
```
?2+2*5
12
READY.
■
```

You can tell PET, however, to do an operation first by enclosing the operators and the numbers in parenthesis:



```
?(2+2)*5
20
READY.
■
```

A good rule to insure accuracy is to use parenthesis whenever you might get confused. PET will take as many levels of parentheses as you care to type in, even to the absurdity of 10 levels in the following example:



```
?((((((((((2+2))))))))))
4
READY.
■
```

Count carefully! . . . You must have as many closing parentheses as you have opening parentheses or PET will say ?SYNTAX ERROR.

Continue to use your PET in this calculator mode. PET basic has a great number of powerful scientific, transcendental, and trigonometric functions which are described in an appendix to this booklet.

Exercise 6 — Using the cursor (... and introduction to screen editing)

By this time you have probably found that (if you are not a typist — or even if you are) it's sometimes challenging, to say the least, to type lines into your PET correctly the first time. Sure, you can use the INST/DEL key to erase the last character typed, as we explained earlier. But what if you typed something wrong at the beginning of the line? You could delete characters back to that point, then retype.

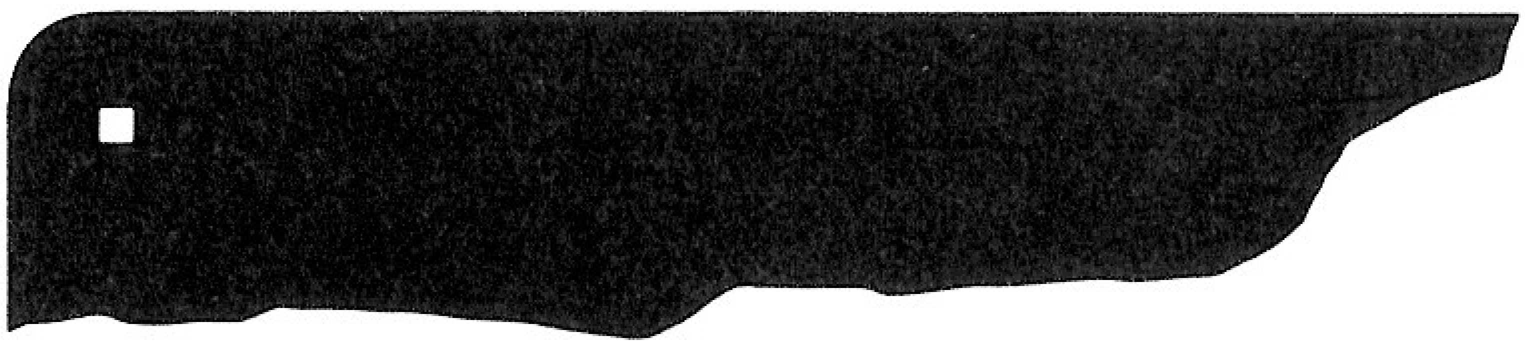
But there is an easier way: PET has a feature called *screen edit* which allows you to move the cursor to any position on the line and at that point either insert, delete, or retype.

> *(The movement of the cursor is non-destructive to the characters over which it passes. The characters will not be deleted or changed as you move the cursor around on the screen.)*

Locate this row of keys on top of the right hand numeric keypad.

CLR HOME   ⇑ CURSOR ⇓   ⇐ CURSOR ⇒   INST DEL

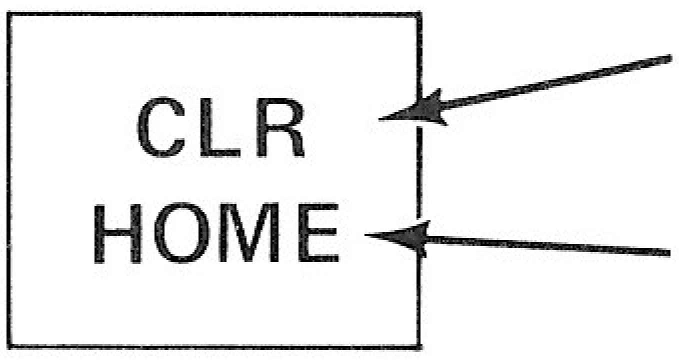
These are all double function keys, their action depending on whether or not a SHIFT key is used. Press the labeled CLR HOME key top. See the cursor move to the top left corner of the screen? This is the "HOME" position.



The same key pressed while the SHIFT key is pressed *clears* the screen. Hold down SHIFT this time and then press CLR HOME.

If there were any characters on the screen, then they were all erased . . . or "cleared."

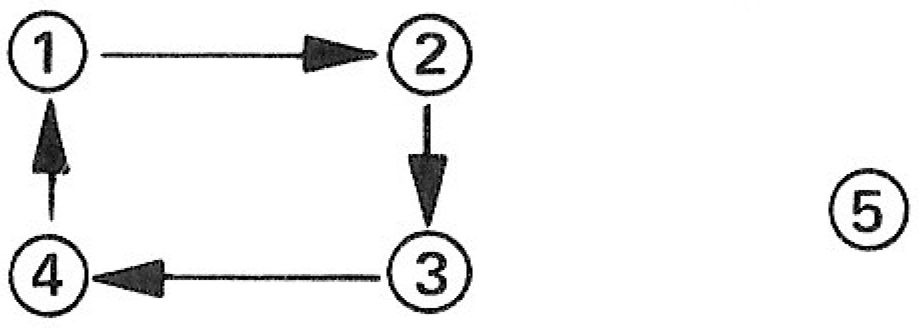
Both functions of this key affect the screen.



With SHIFT – screen is cleared *and* cursor is homed.

Without SHIFT – screen remains same and cursor is homed.

The best exercise to learn the individual cursor movement keys is to move the cursor right, down, left, and up in a sort of circle path to return to the original starting position. You will move the cursor on the screen in a path like this:



To move from ① to ② press the ⇐ CURSOR ⇒ key several times.

For points ② to ③ press ⇑ CURSOR ⇓ . The remaining two sides of the movement require use of the SHIFT Key. Move from ③ to ④ by holding down SHIFT and pressing ⇐ CURSOR ⇒ .

(If you press the last key too many times and wind up in position ⑤ you have discovered another feature called "wrap-around" which has moved the cursor to the end of the previous line. Type ⇐ CURSOR ⇒ without the shift key held down to move the cursor back to position ④ .

The home stretch from ④ to ① is easy. You can either hold down SHIFT and type ⇑ CURSOR ⇓ repeatedly until the cursor is in position 1 or type CLR HOME once to move the cursor "HOME." Try it both ways. Try moving the cursor around the screen between two arbitrary points. Practice until you are confident you can put the cursor where you want it on the screen.

Exercise 7 — Using graphics

If you have accomplished moving the cursor, then you can use your PET like an electronic sketch pad. The characters on the upper half of each keytop are called graphics. When you hold down the SHIFT key as you type, the graphics are printed instead of letters or numbers.

Now let's draw a figure that should look very much like this by the time we get through.

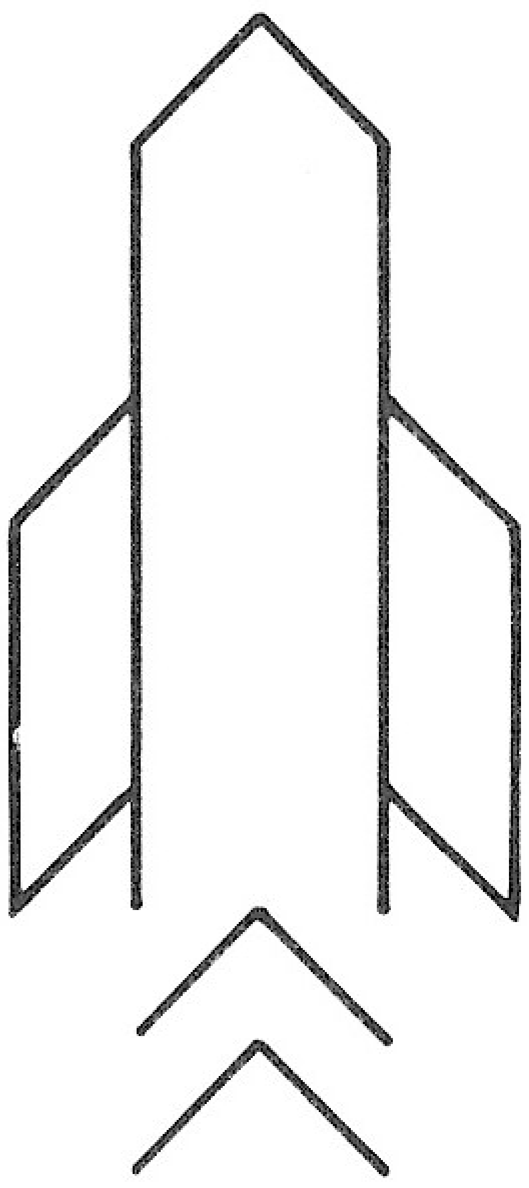


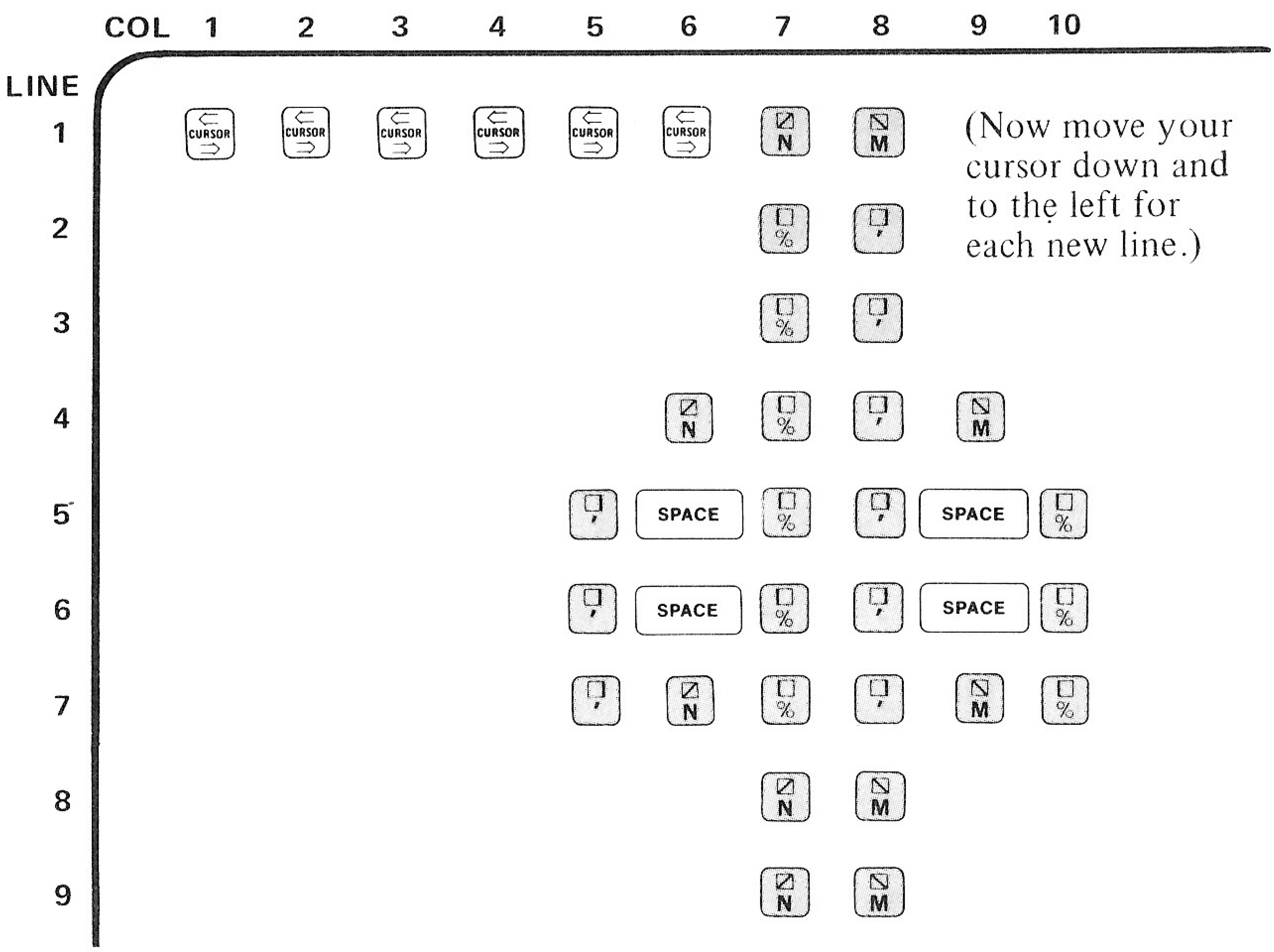
Figure 4. Rocket drawing using the graphics keys

Follow the instructions exactly as shown in the diagram that follows:

Remember the CLR HOME *and* SHIFT *keys? Use these keys now to clear the screen.*

Move the cursor to the right 6 spaces; as shown in the diagram. Press SHIFT *and type the graphics. Now you use the cursor keys to get the cursor in position to type the next line.*

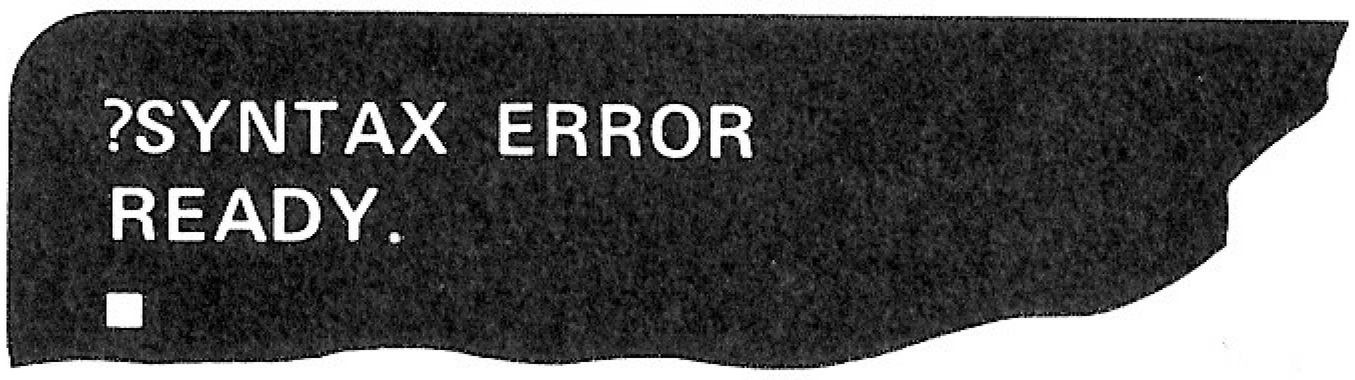
NOTE: Shaded keys are keys that must be accompanied by pressing the SHIFT *key.*



(Now move your cursor down and to the left for each new line.)

Figure 5. Graphics keys used to draw rocket

*NOTE: Do **not** press* RETURN *at any time in this exercise. Your PET will think you've finished; it will not understand and will display:*



```
?SYNTAX ERROR
READY.
■
```

If this happens, first clear the screen again and start over.

Now type N E W RETURN.

The cursor should be in the lower left part of your screen when you've done all the above.

Exercise 8 — Creating a program

When you have finished this exercise, you will have drawn a picture on the screen. You probably went to a lot of work to create this picture. You'd like to preserve it so you can view it again. So let's turn each line of the picture into a program step and see what happens.

> *The importance of a program to a computer can be likened to the importance of a driver to a car. The car does nothing without a driver and the computer does nothing without a program.*

> *A program is stored as a list of steps or instructions in PET's memory. Before we can create a program in its memory, we should make PET forget about any previous program. This is what we did when we typed the word NEW. Use the command any time you want to enter a NEW program.*

Press the CLR HOME key. Make absolutely sure you do ***not*** press SHIFT. You want the cursor in the "HOME" position.

> *(If you were to press* SHIFT*, your PET would **clear** the screen. Awful, after all that work.)*

Now type 1 ? □ " RETURN. The number 1 tells PET "this is the first thing to do." The ? tells PET to print, and the quotes tell PET to print a MESSAGE.

> *(If you make an error, do **not** try to correct it. Instead press* RETURN*, then move the cursor up and type the correct number, the question mark, and the quotes, then press* RETURN.

Now type 2 ? □ " RETURN. The 2 tells PET that this is the second thing to do.

For the third line, type 3 ? □ " RETURN, and for the fourth line, type 4 ? □ " RETURN.

Notice that the only thing that changed has been the number (1, 2, 3 and 4) that tells PET "this is the _____ thing to do." So now tell PET the 5th, 6th, 7th, 8th, and 9th things to do, just like we've done with the first four. Be sure to keep the numbers in the right sequence.

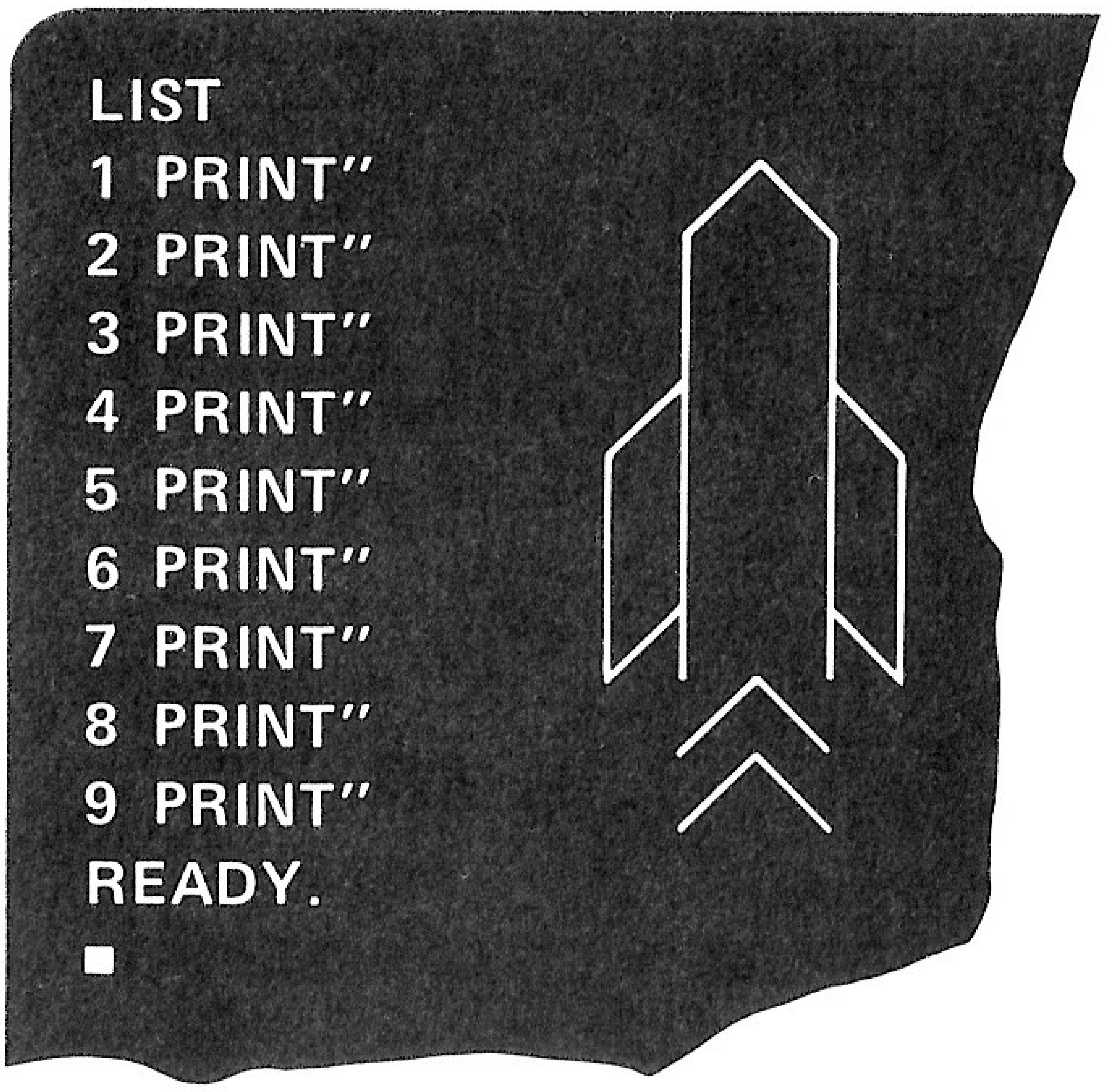
Stop when you reach the line containing the word "NEW," because you don't want that word included in your program. Using the ⇑ CURSOR ⇓ key, move the cursor down the screen until it is below the word "READY." Would you believe you've just created a computer program?

Exercise 9 — Listing and running your program

Clear the screen and type:

LIST RETURN

LIST is a command to your PET to print the lines of program stored in memory onto the screen so that you can look at them. You should see something like this on your screen



```
LIST
1 PRINT"
2 PRINT"
3 PRINT"
4 PRINT"
5 PRINT"
6 PRINT"
7 PRINT"
8 PRINT"
9 PRINT"
READY.
■
```

The ? that you have typed in as a shorthand for PRINT has been expanded out in the listing. Other than that, everything should be as you typed it in. If there is an extra line which should not be there, it may be deleted by typing just the number of the line followed by RETURN.

LIST your program again if you wish. When everything is just as you want it to be type RUN RETURN.

There! Your picture will appear on the screen. RUN tells PET to execute the BASIC program you have entered, starting at the lowest line number step and proceeding with subsequent steps in ascending line number order.

Exercise 10. Amending the program

RUN your program again. If you did not clear the screen first, you may have seen the old rocket disappear at the top of the screen and the new rocket roll up from the bottom of the screen.

This phenomenon is called "scrolling." When PET is printing in the bottom-most line of the screen, everything moves up rather than the cursor moving to a lower line. PET cannot scroll the other way, however. Information that scrolls off the top of the screen is lost.

We can use this scrolling effect to our advantage to produce an animation in which it appears as though a stream of rockets are blasting off from the bottom of the screen and are streaking off the top. To do this we will learn a new BASIC language command.

Type this line in

1 Ø Ø [SPACE] G O T O [SPACE] 1 [RETURN]

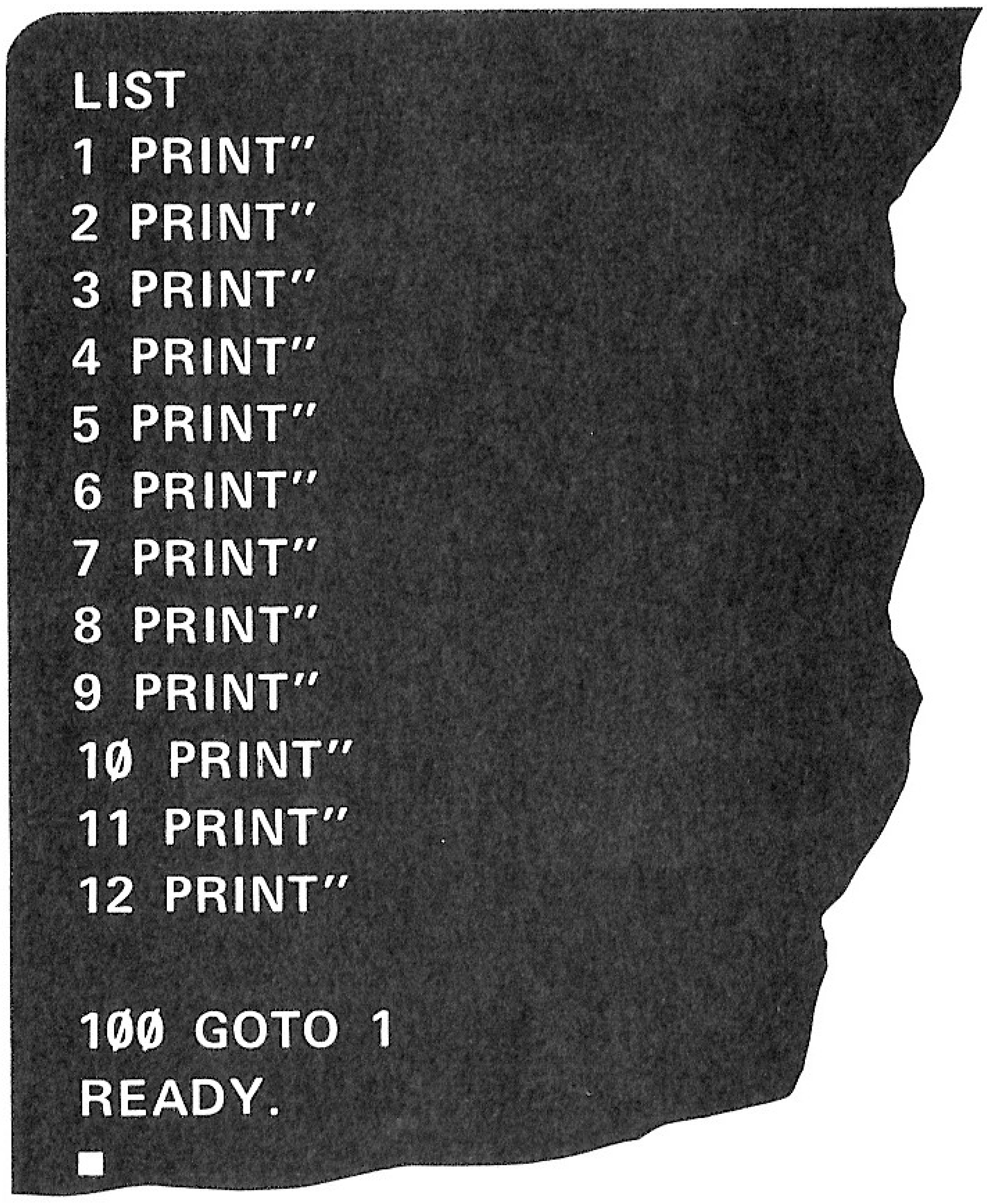
The line number (1ØØ) was chosen so that it would be greater than any you had used previously and thus would be the last step of your program to execute.

GOTO is a BASIC command to break the sequential execution of statements and "go to" the line number specified. If you entered the rocket picture with line numbers exactly as shown, line 1 is the first line of the program which prints the rocket picture. Change the target line of the "go to" to correspond to your first line number if it is not line 1. The effect of line 1ØØ is to repeatedly print the rocket and scroll it off the screen.

But, because we don't want the rockets to be touching nose to tail, we'd like to add some space between them. When we typed LIST, we noticed that the last line number was 9. (We've since added line 1ØØ). Any numbers greater than 9 and smaller than 1ØØ will be positioned correctly in line number sequence by your PET. So let's add the statements:

```
1 Ø ? "
1 1 ? "
1 2 ? "
```

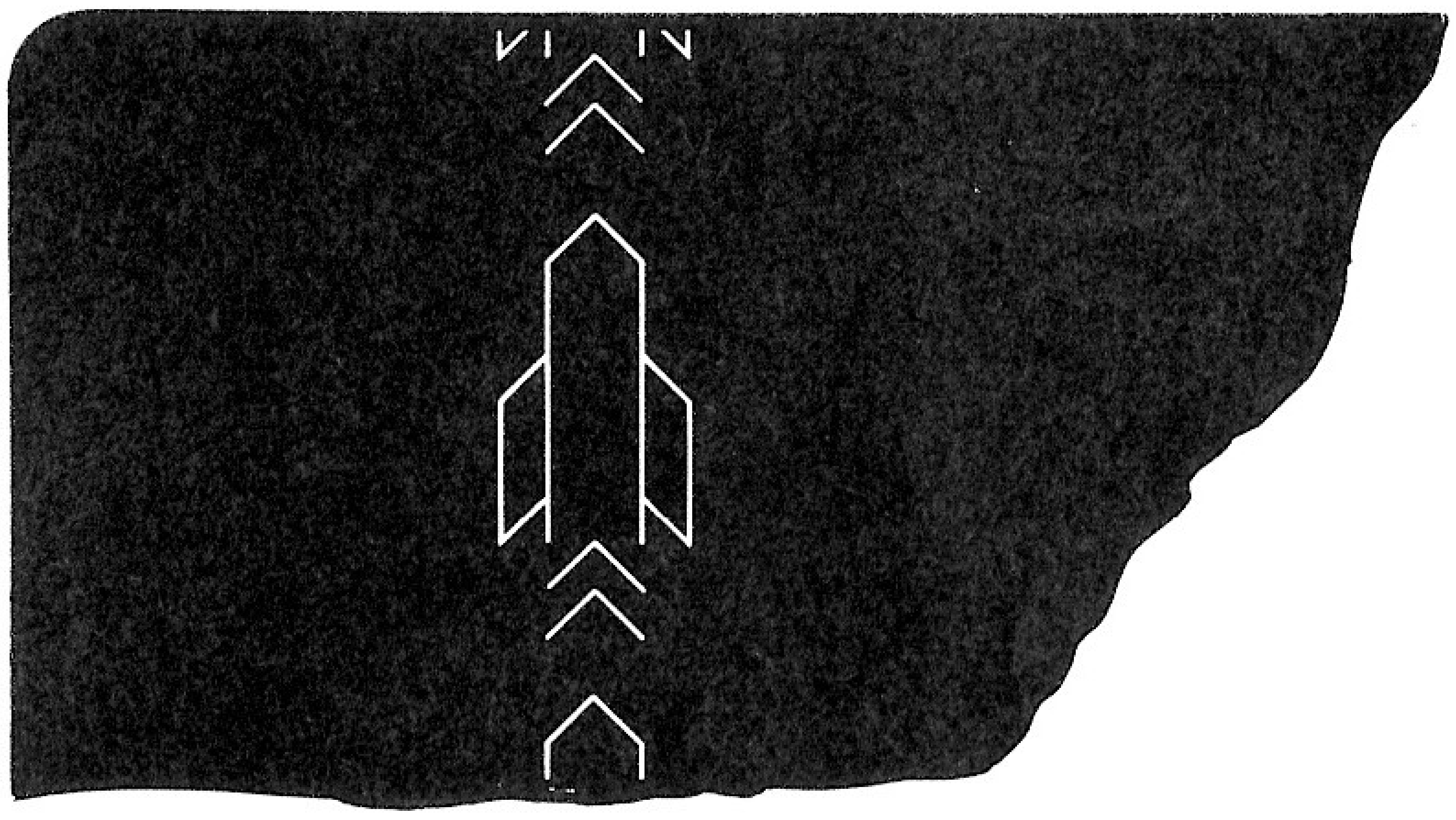
Now let's LIST once more:



```
LIST
1 PRINT"
2 PRINT"
3 PRINT"
4 PRINT"
5 PRINT"
6 PRINT"
7 PRINT"
8 PRINT"
9 PRINT"
1Ø PRINT"
11 PRINT"
12 PRINT"

1ØØ GOTO 1
READY.
■
```

Now type RUN. As soon as you press RETURN PET will execute your program.



Rockets should be flashing on the screen so fast that it may be difficult to see them. The speed at which characters are printed on the screen can be controlled while the program is running by pushing the key OFF RVS . Hold this key down while you watch the screen. Now, release the key. Use of this key reduces the printing speed to about 2 lines per second.

The program you have created contains what is called an "infinite loop." Statement 1ØØ does not contain a condition to stop running the program and cease printing the rocket, but unconditionally goes to the start of the program over and over. It will continue like this forever unless you pull the plug.

(Pulling the plug or shutting off the power-on switch not only stops the program, it also destroys the program statements. You've put in a lot of time typing them and may not want them destroyed.)

PET has a key to press: RUN STOP. This is a STOP function when you do ***not*** press the SHIFT key. This will effectively "pull the plug" on this program, without losing the program statements.

PET will respond with something like:

```
BREAK IN LINE 8
```

This message means that execution of your program was stopped when it reached line 8 (or whatever line it was in your case) because you pressed the RUN STOP key.

Exercise 11 — Screen editing

One of the handiest features of PET is the ability to modify easily the program you have entered, as we have just seen.

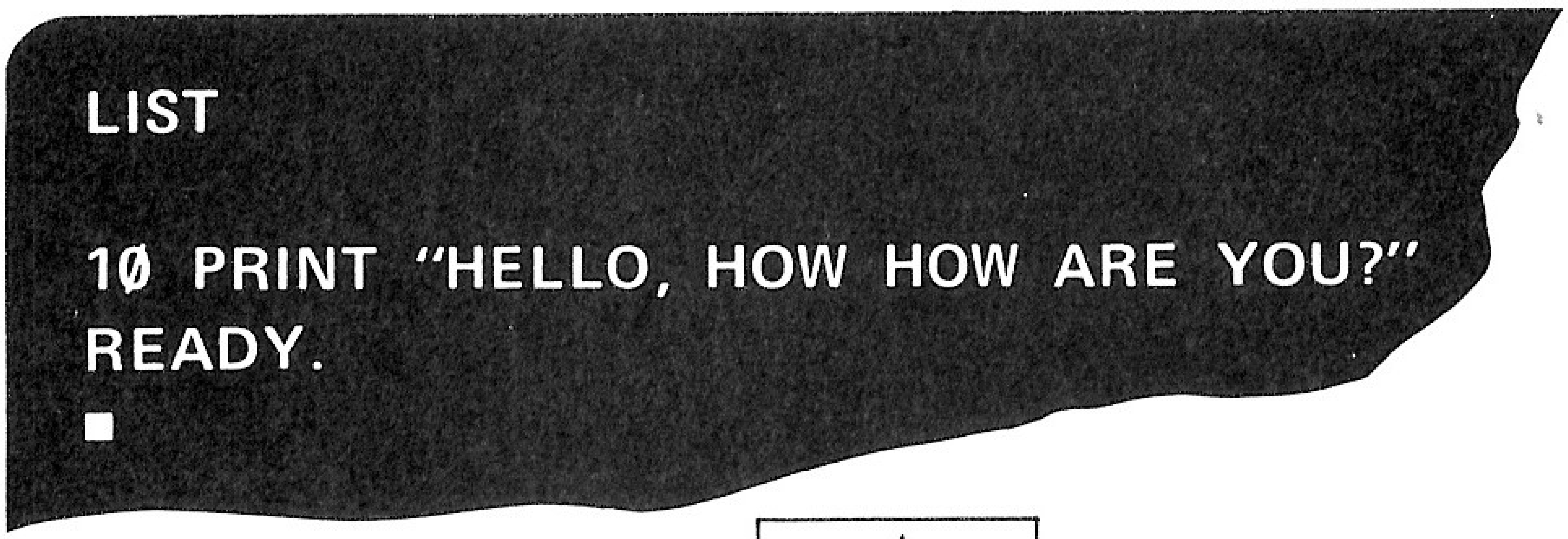
You can change a single character or you can add characters to lines you already have. You can see exactly what you are changing because the changes are visible as you enter them.

Let's try it.

> *(But before we start a new program, let's type* NEW *and press* RETURN*. This is important: it clears all previous programs in your PET and thus avoids any confusion.)*

Type in: **1Ø PRINT "HELLO, HOW HOW ARE YOU?"** RETURN

We have one too many HOWs in the line. Let's type LIST so we can see the line on the screen.



```
LIST

1Ø PRINT "HELLO, HOW HOW ARE YOU?"
READY.
■
```

Now press SHIFT and ⇑ CURSOR ⇓ together. Repeat.

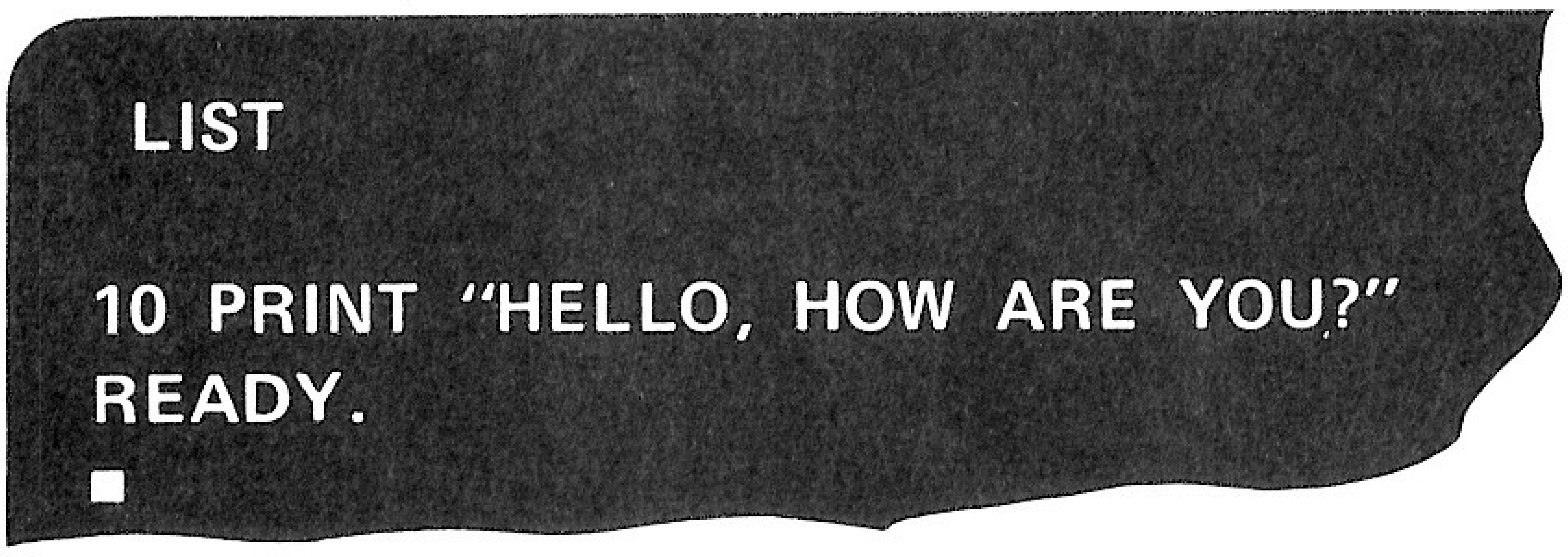
This will move the cursor *up* two lines from its lower position to the first position of line 10.

Now press ⇐ CURSOR ⇒ (without the SHIFT key) several times until it is over the space after the W in either one of the HOWs.

Now press INST DEL 4 times. The extra HOW and a space are gone!

Press RETURN to tell your PET you've finished editing this line.

Now press ⇑ CURSOR ⇓ to get to a blank line on the screen. Type LIST.

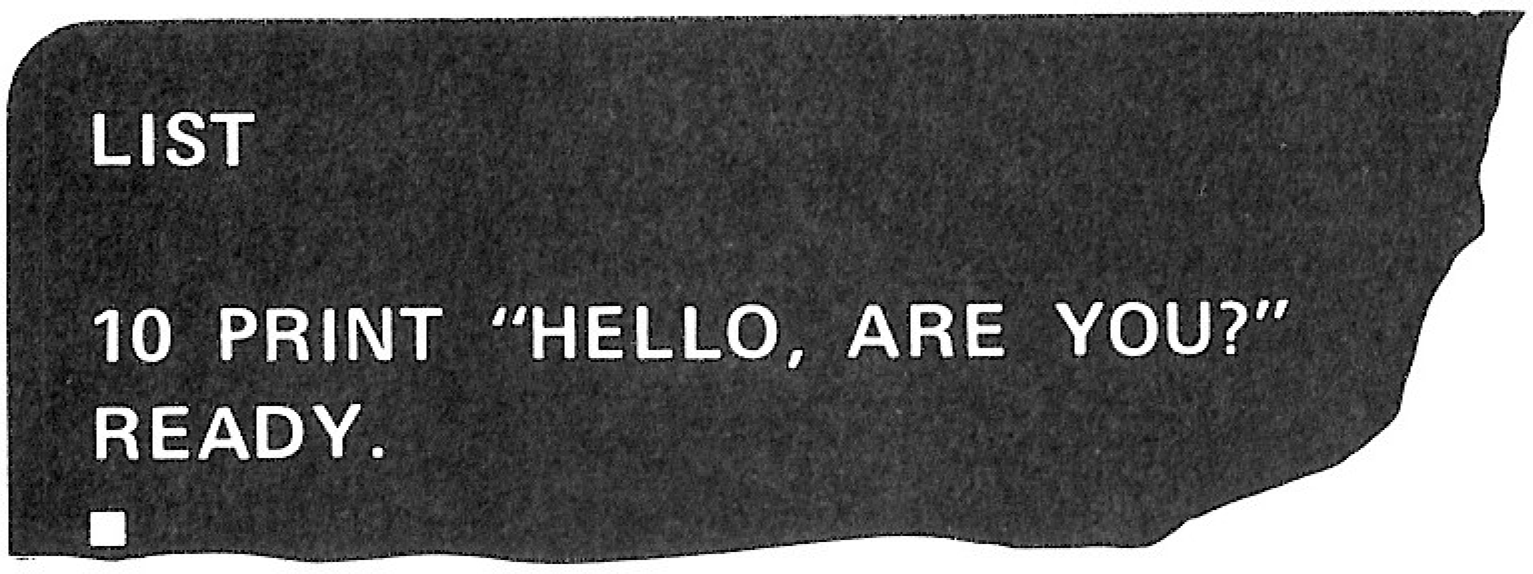


```
LIST

10 PRINT "HELLO, HOW ARE YOU?"
READY.
■
```

You see how easy that was.

Do the same thing again until you have eliminated the HOW:



```
LIST

10 PRINT "HELLO, ARE YOU?"
READY.
■
```

Now let's insert the missing HOW.

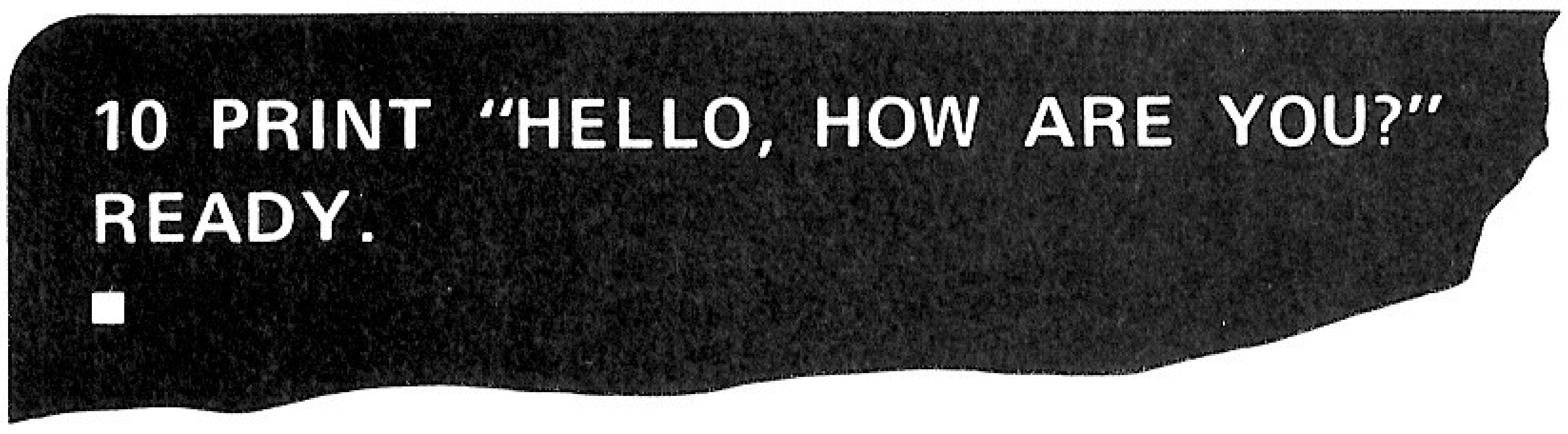
Press SHIFT and CURSOR together. Repeat as before.

Now position the cursor over the A in ARE by pressing CURSOR several times.

Now press SHIFT and INST DEL then H
SHIFT and INST DEL then O
SHIFT and INST DEL then W
SHIFT and INST DEL then SPACE and RETURN

then CURSOR to get past the READY in the display. Type LIST.

You now have:



```
10 PRINT "HELLO, HOW ARE YOU?"
READY.
■
```

With editing that easy you need have no fear of making typing errors. Agreed?

Let's try another interesting screen edit. First, type NEW to clear out the old program.

Enter:

1 0 P R I N T "ANYTHING" RETURN

(This time we won't type LIST each time we make a correction.)

Press SHIFT and CURSOR so that you position the cursor over 1 in line #10.

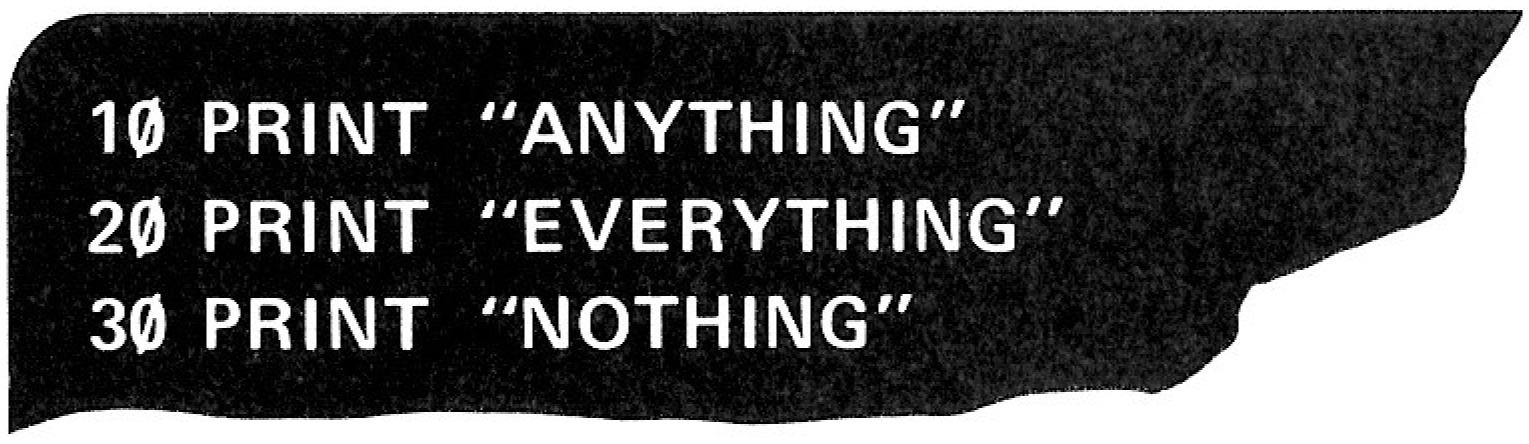
Press 2 then press CURSOR until it reaches the A in ANYTHING"

Now type EVERYTHING" and press RETURN.

Now press SHIFT and CURSOR together so that you position the cursor over the 2 in line #20. Press 3 then CURSOR over to the E in EVERYTHING and type NOTHING". Press SPACE three times. (Because EVERYTHING is three letters longer than NOTHING.) Press RETURN.

Now type LIST and press RETURN.

You'll read:



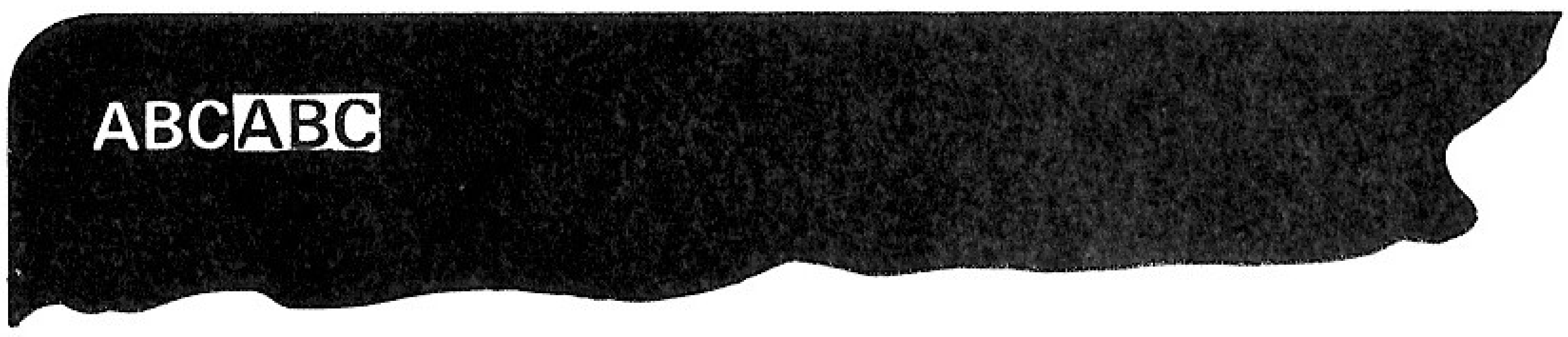
```
10 PRINT "ANYTHING"
20 PRINT "EVERYTHING"
30 PRINT "NOTHING"
```

Interesting? Think of the applications. If you want to repeat a complex statement several times in the same program . . . or if you want to change just a part of a statement on one line and enter that amended statement on another line.

Exercise 12 – Using the reverse field

Every key on the keyboard, with the exception of a few which we shall note, prints almost exactly what you see onto the screen. We say "almost" because the screen displays characters in white on a black background. There is a OFF RVS key which, when pressed, causes all subsequent characters to be displayed in reverse field – black on white – on that line.

Type A B C OFF RVS A B C and you'll see:



```
ABCABC
```

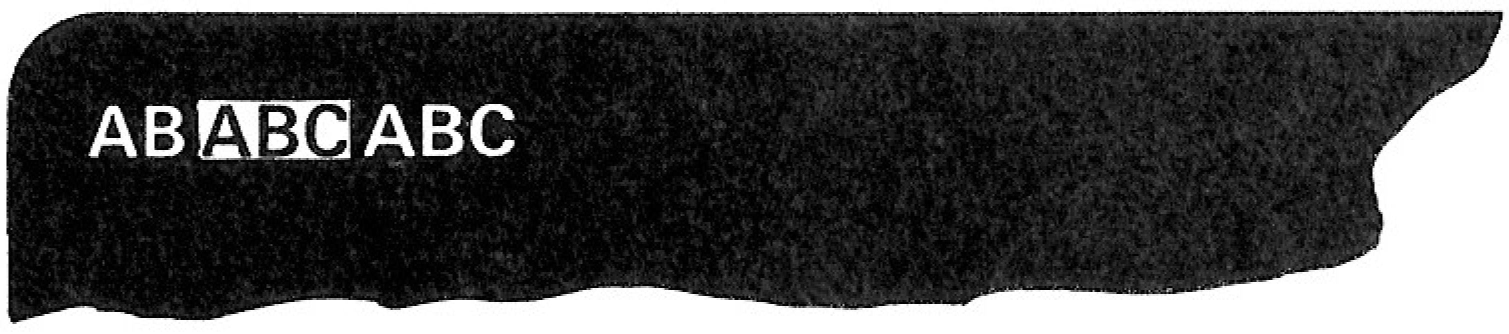
Your PET displays 128 unique symbols which, with the addition of reverse field, really adds up to a total of 256 different characters that can be displayed.

Reverse field remains in effect until a) you type RETURN or b) you hold down the SHIFT and type OFF RVS.

As an example, type:

A B OFF RVS A B C SHIFT OFF RVS A B C

You'll see:



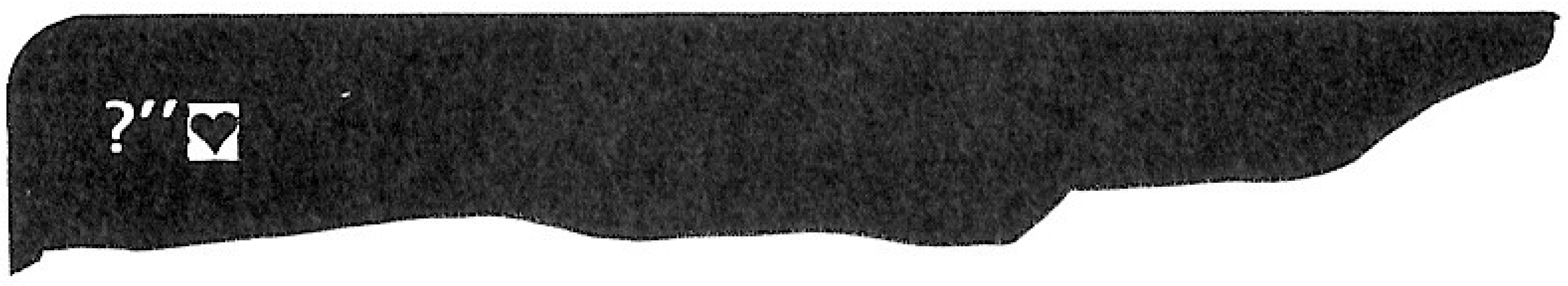
```
ABABCABC
```

Exercise 13 — Programming cursor movement

Cursor control characters may be programmed into PRINT statements. It is often desirable to clear the PET display under program control. We will do it in a direct statement.

? " SHIFT CLR HOME

Note that you did not clear the screen by typing these keys, but that a reverse field heart appeared on the screen.



```
?"♥
```

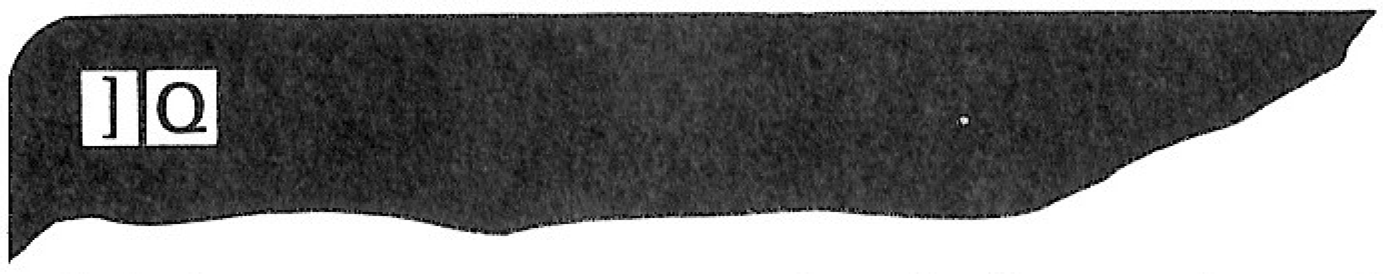
When you have typed an odd number of quote marks you are in this special cursor control character insertion mode.

(" Represents a *single* quote mark, for this discussion. And one is an odd number.)

The ♥ is a representation of a CLEAR SCREEN control character. Do not type RETURN yet. Instead type:

⇐ CURSOR ⇒   ⇑ CURSOR ⇓

These print out as



```
]Q
```

which are cursor control characters for CURSOR RIGHT and CURSOR DOWN.

If you now type a second " you will have entered an even number of quote marks and you will leave the special mode. Typing ⇐ CURSOR ⇒ will again move the cursor, but this time, without printing anything.

Any time you want to *enter* or *leave* the control character insertion mode you may do one of two things:

1. Enter a second " , press the RETURN key. Then use your cursor keys to return to the point on the preceding line . . . or
2. Delete the first quotation: " DEL.

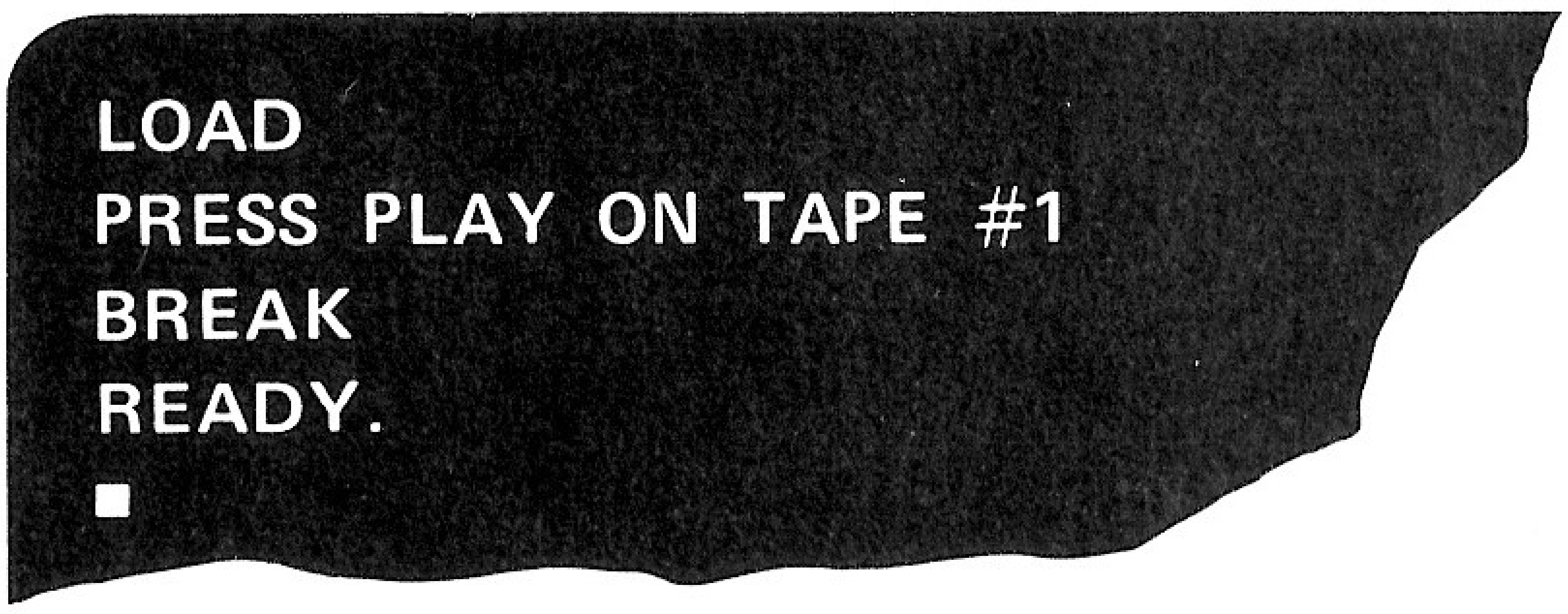
Using your cassette to load a program

The built-in cassette drive in your PET computer is your easy access to a library of BASIC programs, either created by you or purchased from the extensive COMMODORE library.

PET is like a pocket calculator in that it forgets everything when you turn the power off. (Remember what happens to time?) That is why PET has a built-in cassette drive. Programs can be saved on tape before power is turned off. They may be restored to PET's memory when power is turned on again.

Take a cassette, hit "Eject" on the cassette drive to open the cover, and place the cassette in just as you would a normal audio cassette. Do not push any cassette keys at this time.

Now, hold down the SHIFT key and touch and release the RUN STOP key. If you see:

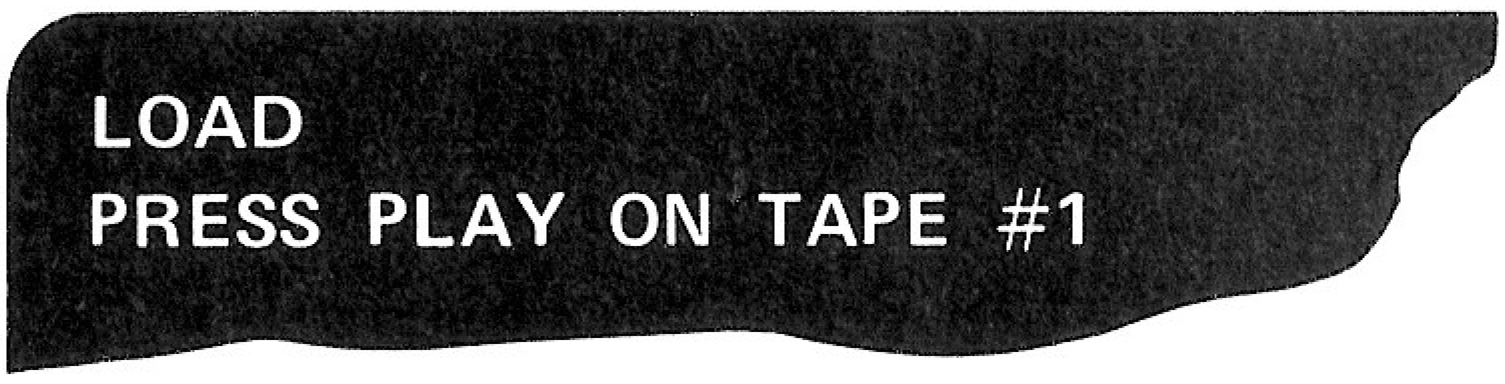


```
LOAD
PRESS PLAY ON TAPE #1
BREAK
READY.
■
```

then you've released the SHIFT key *before* you released the RUN STOP key. Don't fret . . . just try again.

Hold the SHIFT key down, touch and release the RUN STOP key. *Now* you can release the SHIFT key.

If you've done all this correctly, you should see:



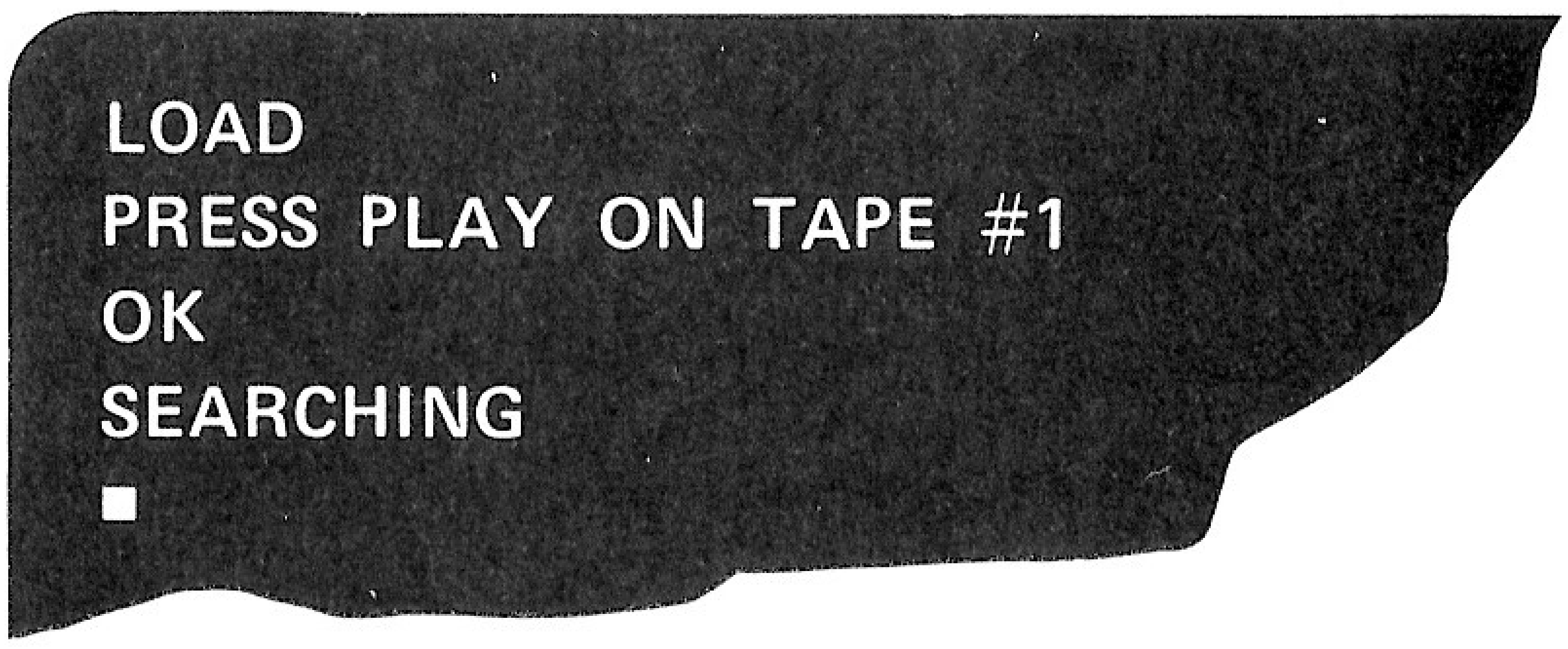
```
LOAD
PRESS PLAY ON TAPE #1
```

Pressing the RUN STOP and the SHIFT keys caused the command LOAD to be typed on the screen and PET responded by asking you to operate the cassette.

(If you wish, you can also tell your PET to load the program by typing in LOAD and RETURN *instead of the above procedure.)*

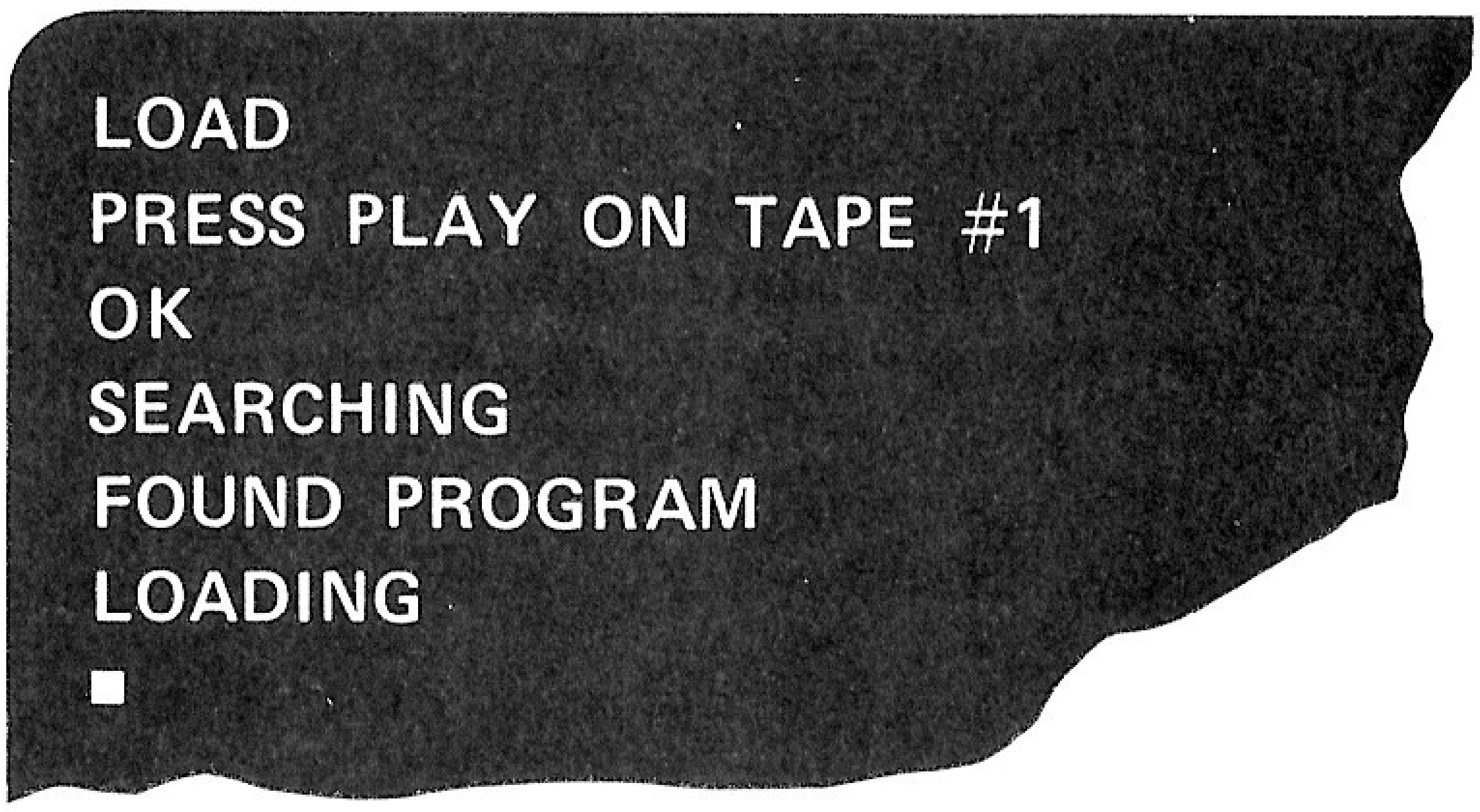
Press the key labeled "PLAY" on the cassette unit.

Your screen should now display:



```
LOAD
PRESS PLAY ON TAPE #1
OK
SEARCHING
■
```

This means your pressing of the key is acknowledged and PET is now searching for data on the tape. In a few seconds (about 5-10) you will see:



```
LOAD
PRESS PLAY ON TAPE #1
OK
SEARCHING
FOUND PROGRAM
LOADING
■
```

PET has found a program on the tape and is transferring it from the cassette into its memory. This is the operation referred to as "loading."

When loading is complete, the program will automatically begin executing. Also, the cassette motor will be turned off.

(We're assuming that your PET's program is labeled "PROGRAM" for the sake of this example. It could be labeled virtually anything else.)

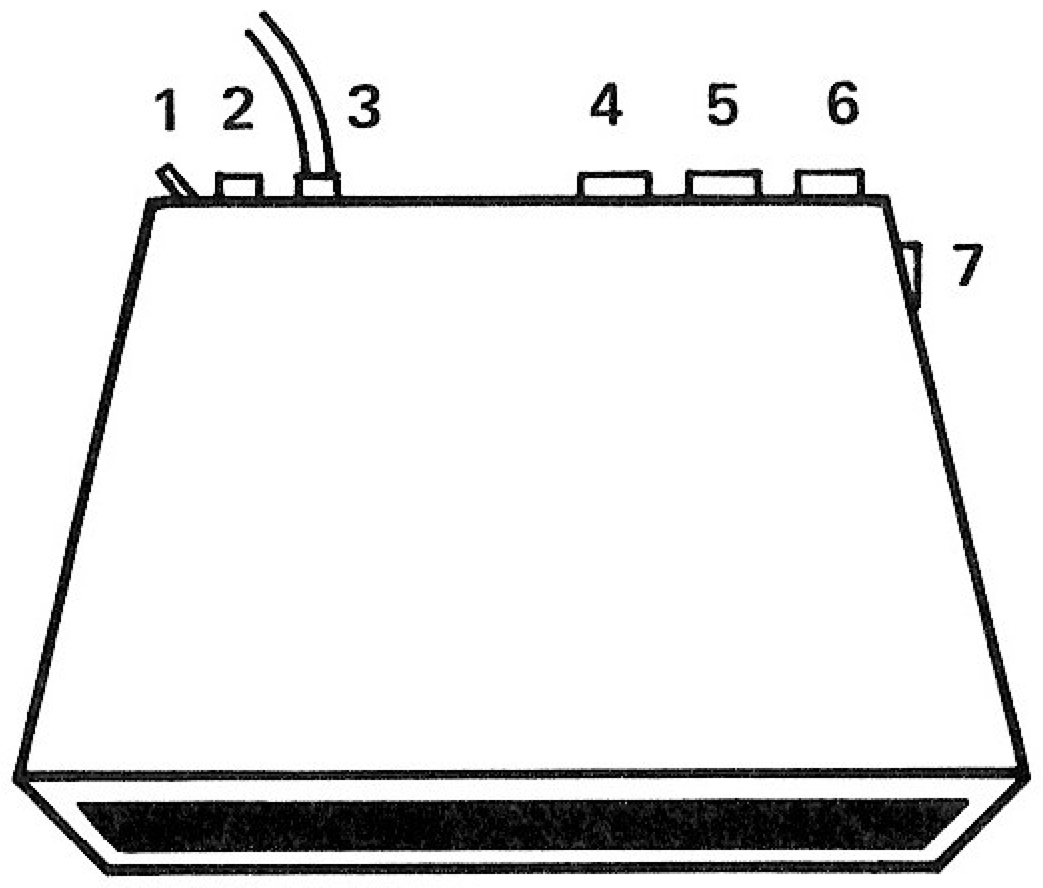
Disclaimer on Software:

The complex and extensive software of the PET computer has been thoroughly tested and is believed to be entirely reliable. However, no responsibility is assumed by Commodore or your sales agent for inaccuracies.

Appendix

I. Interfaces

PET INTERFACES AND LINES TO THE OUTSIDE WORLD



① Power switch
② 1.6 amp SLO-BLO fuse
③ 120 VAC with ground line cord
④ IEEE-488 interface
⑤ Parallel user port
⑥ 2nd cassette interface
⑦ Memory expansion connector

Figure 6. Overhead view of the PET computer

IEEE-488 INTERFACE

The IEEE-488 Interface is brought out on the PET main logic PC board to 12 pins on each side of the board. This differs from the IEEE-488 standard which calls out:

> *"The Microribbon (Amphenol or Cinch Series 57) or Champ (Amp) connectors may be used for this application."*

This refers to:	*Cinch*	*571Ø24Ø*	*Solder-plug*
	Cinch	*572Ø24Ø*	*Solder-receptacle*
	Amp	*5523Ø1-1*	*Insulation displacement plug*
	Amp	*5523Ø5-1*	*Insulation displacement receptacle*

These above-cited connectors are not directly connectable to the PET logic board. We suggest the list of more common receptacles that do connect to the PET logic board, as shown.

12 positions, 24 contacts edge card connector with .156" spacing. Similar pinout as standard IEEE connector. Keyed between pins 2-3 and 9-10.

	PET	IEEE			PET	IEEE		
TOP	1	1	DIO1	BOTTOM	A	13	DIO5	
	2	2	DIO2		B	14	DIO6	
	3	3	DIO3		C	15	DIO7	
	4	4	DIO4		D	16	DIO8	
	5	5	EIO		E	17		
	6	6	DAV		F	18		
	7	7	NRFD		H	19		
	8	8	NDAC		J	20		⏚
	9	9	IFC		K	21		Ground
	1Ø	1Ø	SRQ		L	22		
	11	11	ATN		M	23		
	12	12	Chassis ground		N	24		

TYPICAL CONNECTORS

Sylvania
6AGØ1-12-1A1-Ø1
Amp
53Ø657-3
Amp
53Ø658-3
Amp
53Ø654-3
Cinch
251-12-9Ø-16Ø

PARALLEL USER PORT

12 positions, 24 contacts edge card connector with .156" spacing. Keyed between pins 1-2 and 1Ø-11.

TOP	1	⏚ Ground	BOTTOM	A	⏚ Ground
	2	T.V. Video		B	CA1
	3	IEEE SRQ		C	PAØ
	4	IEEE EOI		D	PA1
	5	Diagnostic Sense		E	PA2
	6	Tape #1 read		F	PA3
	7	Tape #2 read		H	PA4
	8	Tape write		J	PA5
	9	T.V. Vertical		K	PA6
	1Ø	T.V. Horizontal		L	PA7
	11	⏚ Ground		M	CB2
	12	⏚ Ground		N	⏚ Ground

TYPICAL CONNECTORS

(See IEEE-488 on page 22)

Register addresses: (See MOS 6522 specification)

$E841	Data (with handshake)
$E84F	Data (without handshake)
$E843	Data direction register
$E84B	Auxiliary control register
$E84C	Peripheral control register

(NOTE: These are hexidecimal addresses. Convert to decimal if used in BASIC.)

2nd CASSETTE INTERFACE

6 positions, 12 contacts edge card connector with .156" spacing. Keyed between pins 2-3.

TOP	1	BOTTOM	A	GND
	2		B	+5
	3		C	Motor
	4		D	Read
	5		E	Write
	6		F	Sense

TYPICAL CONNECTORS

Sylvania	*6AJØ7-6-1A1-Ø1*
Viking	*2KH6/1AB5*
Viking	*2KH6/9AB5*
Viking	*2KH6/21AB5*
Amp	*53Ø692-1*
Sullins	*ESM6-SREH*
Cinch	*25Ø-Ø6-9Ø-17Ø*

MEMORY EXPANSION

40 positions, 80 contacts edge card connector with .1" spacing. No keys. Top side at connector (B1-40) is grounded.

A1	AØ	A11	A10	A21	$\overline{\text{SEL}}$ 6	A31	NC
A2	A1	A12	A11	A22	$\overline{\text{SEL}}$ 7	A32	NC
A3	A2	A13	NC	A23	$\overline{\text{SEL}}$ 9	A33	BDØ
A4	A3	A14	NC	A24	$\overline{\text{SEL}}$ A	A34	BD1
A5	A4	A15	NC	A25	$\overline{\text{SEL}}$ B	A35	BD2
A6	A5	A16	$\overline{\text{SEL}}$ 1	A26	NC	A36	BD2
A7	A6	A17	$\overline{\text{SEL}}$ 2	A27	$\overline{\text{RES}}$	A37	BD4
A8	A7	A18	$\overline{\text{SEL}}$ 3	A28	$\overline{\text{IRQ}}$	A38	BD5
A9	A8	A19	$\overline{\text{SEL}}$ 4	A29	$\overline{\text{BØ2}}$	A39	BD6
A1Ø	A9	A2Ø	$\overline{\text{SEL}}$ 5	A3Ø	R/$\overline{\text{W}}$	A4Ø	BD7

TYPICAL CONNECTORS

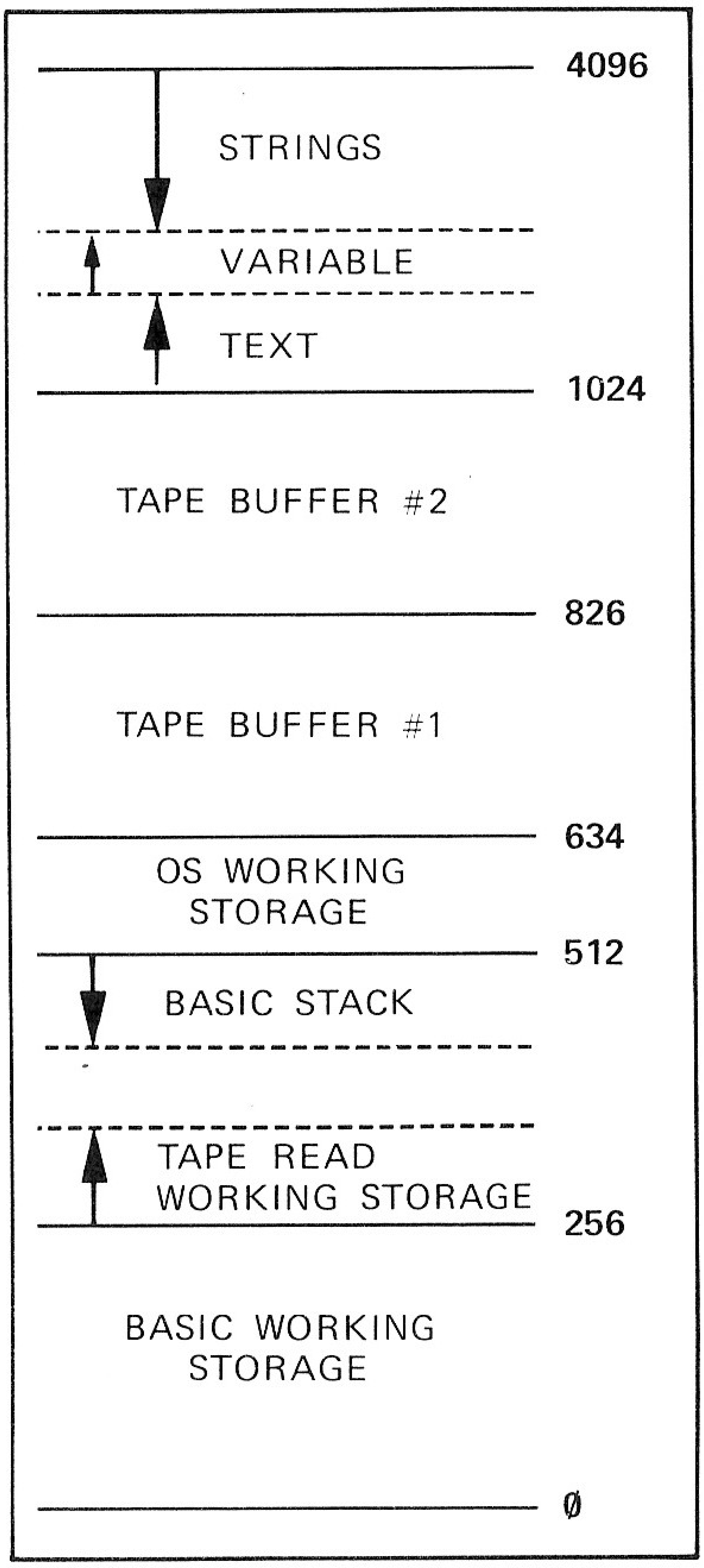
Sylvania
6ADØ-4Ø-1A1-Ø1
Viking
3KH4Ø/1JN5
3KH4Ø/9JN5
Sullins
ESC-4Ø-DREH
Cinch
251-4Ø-3Ø-41Ø

Address selects are decoded in 4K blocks, i.e., $\overline{\text{SEL}}$ 1 selects $1ØØØ-IFFF, $\overline{\text{SEL}}$ B selects $BØØØ-BFFF.

PET MEMORY MAP (IN 4K BLOCKS)

F	I/O, Diagnostics, Monitor ROM		
E	$E8ØØ-	I/O Ports and Expansion I/O	
	$EØØØ-E7FF	Screen Editor ROM	
D C	Basic ROM		
B A 9	Expansion ROM		
8	$8000-$83E7	TV display	RAM
7			
6 5 4 3 2	Expansion RAM		
1	Basic Text RAM (8k Version)		
Ø	Page Ø	$A-$5A	Basic Input Buffer
	Page 1	Stack	
	Pages 2-3	$2ØØ	3 byte clock register
		$219	Interrupt vector
		$21B	Break inst. vector
		$27A	Buffer for cassette #1
		$33A	Buffer for cassette #2
	Pages 4-8	$400-$FFF	Basic Text RAM

II. RAM usage by basic interpreter and operating system



If the cassette units are not used, the user may then insert machine code subroutines into the buffer areas used to support these devices.

If cassette #2 *is not* used, then 192 bytes are available starting at 826. If *no* cassettes are used, storage is available for 192 bytes, beginning at 634, and another 192 bytes at location 826.

Text and variables occupy locations above 1024 when the interpreter is running.

Strings storage is dynamic and moves downward from the end of memory.

By taking these latter characteristics into account, it is possible to utilize maximum storage *above* the variables and *below* string storage.

III. Basic commands

Basic Commands and Statements

COMMAND/ STATEMENT	EXAMPLE	PURPOSE
CLR	CLR	Sets variables to zero or null.
CMD	CMD D	Keep IEEE device D open to monitor bus.
CONT	CONT	Continue program execution after a STOP command. No program changes permitted.
GOTO	GOTO L	Continue program execution at line L after a STOP command. Program changes are permitted.
FRE	PRINT FRE (0)	Returns number of bytes of available memory.

Basic Commands and Statements (Continued)

COMMAND/ STATEMENT	EXAMPLE	PURPOSE
LIST	LIST	Lists current program.
	LIST -L	Lists current program through line L.
	LIST L-M	Lists lines L through M of current program.
	LIST L-	Lists current program from line L to end.
LOAD	LOAD	Loads next encountered program from built-in tape unit.
	LOAD "NAME"	Loads file NAME from built-in tape unit.
	LOAD "NAME," D	Loads file NAME from device D.
NEW	NEW	Deletes current program from memory, sets variables to zero.
PEEK	PEEK(A)	Returns byte value from address A.
POKE	POKE A,B	Loads byte B into address A.
PRINT	PRINT A	Prints value of A on display screen.
	PRINT A$	Prints specified string on screen.
	PRINT #D,A	Prints value of A on device D.
	PRINT #D,A$	Prints specified string on device D.
RUN	RUN	Begins execution of program at lowest line number.
	RUN L	Begins execution of program at line L.
SAVE	SAVE	Saves current program on built-in tape unit.
	SAVE "NAME"	Saves current file or program NAME on built-in tape unit.
	SAVE "NAME," D	Saves current program or file NAME on device D.
	SAVE "NAME," D,C	Saves file NAME on device D. C specifies EOF or EOT.
STOP	STOP	Stops program execution.
SYS	SYS(X)	Complete control of PET is transferred to a subsystem at decimal address contained in the argument.
TI$	TI$="HHMMSS"	Sets PET's internal clock to real time.
	PRINT TI	Displays number of 'jiffies' since PET was powered up or clock was zeroed. (A jiffy = 1/60 of a second.)
USR	USR(X)	Transfers program control to a program whose address is at locations 1 and 2. X is a parameter passed to and from the machine language program.
WAIT	WAIT A,B,C	Stops execution of BASIC until contents of A, ANDed with B and exclusive ORed with C, is not equal to zero. C is optional and defaults to zero.
CLOSE	10 CLOSE N	Closes logical file N.

Basic Commands and Statements (Continued)

COMMAND/ STATEMENT	EXAMPLE	PURPOSE
DATA	10 DATA 1,2,3,4 20 DATA TOM,SUE	Specifies data to be read from left to right. Alphabetics do not need to be enclosed in quotes.
	30 DATA "TOM DOE"	If strings contain spaces, commas, colons, or graphic characters, the string must be enclosed in quotes.
DIM	10 DIM A(n)	Specifies maximum number of elements in an array or matrix.
	20 DIM A(n,m,o,p)	Specifies maximum number of dimensions in an array.
	30 DIM A(n),B(m)	Number of arrays limited by memory.
	40 DIM A(N)	May be dimensioned dynamically.
	50 DIM A$(n)	Strings may be dimensioned.
END	999 END	Terminates program execution.
GET	10 GET C	Accepts single character from keyboard.
	20 GET C$	Accepts single string character from keyboard.
	30 GET #D,C	Accepts single character from specified logical file.
	40 GET #D,C$	Accepts specified single string character from logical file.
INPUT	10 INPUT A	Accepts value of A from keyboard.
	20 INPUT A$	Accepts value of string variable A from keyboard. The string does not have to be enclosed in quotes.
	30 INPUT A,A$,B,B$	Accepts specified values from keyboard.
	40 INPUT #D,A	Accepts value of A from logical file D.
	50 INPUT #D,A$	Accepts specified string from logical file D.
	60 INPUT #D,A,A$,B,B$	Accepts specified values and strings from logical file D. Strings do not have to be enclosed in quotes.
LOAD	10 LOAD	Loads next encountered program or file, on built-in tape unit, into PET's memory.
	20 LOAD "NAME"	Loads program or file NAME into memory from built-in tape unit.
	30 LOAD "NAME",D	Loads specified file NAME from device D.
OPEN	10 OPEN A	Opens logical file A for read only from built-in tape unit.
	20 OPEN A,D	Opens logical file A for read only from device D.
	30 OPEN A,D,C	Opens logical file A for command C from device D.
	40 OPEN A,D,C,"NAME"	Opens logical file A on device D. If device D accepts formatted files, file NAME is positioned for command.
POS	10 PRINT POS(0)	Prints next available print position (position of cursor on screen).
PRINT	10 PRINT A	Prints value of A on display screen.
	20 PRINT A$	Prints specified string on screen.
	30 PRINT A,A$	Prints specified values or strings on screen, beginning in next available print position (pre-TABbed positions are in columns 10,20,30,40, etc.).

Basic Commands and Statements (Continued)

COMMAND/ STATEMENT	EXAMPLE	PURPOSE
	40 PRINT A;A$	Prints on specified values and strings on screen separated by 3 spaces if numeric, concatenated if string.
	50 PRINT #D,A	Prints specified value on logical file D.
	60 PRINT #D,A$	Prints specified string on logical file D.
READ	10 READ A	Obtains value of A from a DATA statement.
	20 READ A$	Obtains string value of A from a DATA statement.
	30 READ A,A$,B,B$	Obtains specified values for strings and numeric variables from DATA statements.
REM	10 REM **COMMENT**	Inserts non-executable comments in a program for documentation purposes.
RESTORE	10 RESTORE	Permits re-reading of DATA statements without re-running program.
TAB	10 PRINT TAB(N);A	Prints value of A in character position N+1 on screen.
	20 PRINT TAB(N);A$	Prints string beginning in character position N+1 on screen.
VERIFY	10 VERIFY	Verifies most recent program saved on built-in cassette by reading it and comparing it with program still in PET's memory.
	20 VERIFY "NAME"	Verifies specified file NAME saved on built-in cassette by reading it and comparing it with program still in PET's memory.
	30 VERIFY "NAME",D	Verifies specified file NAME saved on device D by reading it and comparing it with program still in PET's memory.
SPC	10 SPC(N)	Prints N spaces or blanks.
FOR . . . NEXT	10 FOR A = 1 TO 20 ⋮ 90 NEXT A	Loop control. Performs all instructions between FOR and NEXT as many times as specified by index. In this example, the index variable is A.
STEP	10 FOR A = 1 TO 20 STEP 2 ⋮ 90 NEXT A	Step specifies size of increment to be added to index to increase or decrease its value towards the desired number of iterations.
IF . . . THEN	10 IF A = 10 THEN PRINT A	If condition is 'TRUE,' instruction following 'THEN' (in this example, 'PRINT A') would be executed. Otherwise, the next statement in sequence is executed.
IF . . . GOTO	10 IF A=1 GOTO L	If condition is true, control is transferred to specified line. Otherwise, the next statement, following the IF . . . GOTO, is executed.

Basic Commands and Statements (Continued)

COMMAND/ STATEMENT	EXAMPLE	PURPOSE
GOTO	10 GOTO L	Transfers control (jumps) to specified line, skipping over intervening lines.
GOSUB	10 GOSUB L	Begins execution of a subroutine which begins on a specified line.
ON . . . GOTO	10 ON A GOTO L,M,N	Transfers control to specified line (in this example, L,M, or N, depending on value of index A.
ON . . . GOSUB	10 ON A GOSUB L,M,N	Begins execution of subroutine which begins on line L,M, or N, depending on the value of index A.
RETURN	9990 RETURN	Subroutine exit; transfers control to the statement following most recent GOSUB directing transfer to the subroutine.

String Functions

FUNCTION	EXAMPLE	PURPOSE
ASC	10 A=ASC("XYZ")	Returns integer value corresponding to ASCII code of first character in string.
CHR$	10 A$=CHR$(N)	Returns character corresponding to ASCII code number.
LEFT$	10 ?LEFT$(X$,A)	Returns leftmost A characters from string.
LEN	10 ?LEN(X$)	Returns length of string.
MID$	10 ?MID$(X$,A,B)	Returns B characters from string, starting with the Ath character.
RIGHT$	10 ?RIGHT$(X$,A)	Returns rightmost A characters from string.
STR$	10 A$=STR$(A)	Returns string representation of number.
VAL	10 A=VAL(A$) 20 A=VAL("A")	Returns numeric representation of string. If string not numeric, returns "Ø".

ASC, LEN and VAL functions return numerical results. They may be used as part of an expression. Assignment statements are used here for examples only; other statement types may be used.

Arithmetic Functions

FUNCTION	EXAMPLE	PURPOSE
ABS	10 C=ABS(A)	Returns magnitude of argument without regard to sign.
ATN	10 C=ATN(A)	Returns arctangent of argument. C will be expressed in radians.
COS	10 C=COS(A)	Returns cosine of argument. A must be expressed in radians.
DEF FN	10 DEF FNA(B)=C*D	Allows user to define a function. Function label A must be a single letter; argument B is a dummy.

Arithmetic Functions (Continued)

SYMBOL	EXAMPLE	PURPOSE
EXP	10 C=EXP(A)	Returns constant 'e' raised to power of the argument. In this example, e^A.
INT	10 C=INT(A)	Returns largest integer less than or equal to argument.
LOG	10 C=LOG(A)	Returns natural logarithm of argument. Argument must be greater than or equal to zero.
RND	10 C=RND(A)	Generates a random number between zero and one. If A is less than 0, the same random number is produced in each call to RND. If A = 0, the same sequence of random numbers is generated each time RND is called. If A is greater than 0, a new sequence is produced for each call to RND.
SGN	10 C=SGN(A)	Returns -1 if argument is negative, returns 0 if argument is zero, and returns +1 if argument is positive.
SIN	10 C=SIN(A)	Returns sine or argument. A must be expressed in radians.
SQR	10 C=SQR(A)	Returns square root of argument.
TAN	10 C=TAN(A)	Returns tangent of argument. A must be expressed in radians.

Arithmetic Operators

SYMBOL	EXAMPLE	PURPOSE
=	10 A=B 20 LET A=B	Assigns a value to a variable. Let is optional.
↑	30 PRINT A↑2	Exponentiation; in example, A^2.
/	35 C=A/8	Division
*	40 C=A*8	Multiplication
+	50 C=A+8	Addition
-	60 C=A-8	Subtraction
=	10 IF A=B THEN PRINT C	A 'equals' B.
<>	10 IF A<>B THEN C=4	A 'does not equal' B.
<	10 IF A<B THEN C$="X"	A 'is less than' B.
>	10 IF A>B THEN C$=D$+E$	A 'is greater than' B.
<=	10 IF A<=B THEN C=20	A 'is less than or equal to' B.

Arithmetic Operators (Continued)

SYMBOL	EXAMPLE	PURPOSE
>=	10 IF A>=B THEN C=D-1	A 'is greater than or equal to' B.
AND	10 IF A AND B THEN C=0	A and B must BOTH be true for statement 10 to be true.
OR	20 IF A OR B THEN C=90	A must be true or B must be true for statement 20 to be true.
NOT	30 IF NOT A THEN PRINT C	Expression is true if A is false.

**NOTE: The numerical values used in the evaluation of logical comparisons are: 'TRUE' is any non-zero number and 'FALSE' is zero.

Special Symbols, Commands and Statements

SYMBOLS, COMMANDS, STATEMENTS	EXAMPLE	PURPOSE
:	10A=1:B=2:C=3	Allows multiple statements on a line.
;	10PRINT A;B	Allows same line printing. Elements are separated by 3 spaces.
	20PRINT A$;B$	Allows same line printing. String elements are concatenated.
,	10PRINT A,B	Allows same line printing. Elements are separated and printed in pre-TABbed print positions (columns 10,20,30, etc.)
,	LOAD "NAME," D	Separates elements in LOAD, SAVE, OPEN, and VERIFY.
?	10?A	Abbreviation for PRINT. Stores as one character; lists as word PRINT.
$	10A$="ABCDEFG"	String identifier.
%	10A%=INT(X)	Integer identifier.
"	10A$="ABCDEF"	String enclosures.
carriage return		Must follow every command, statement, or data entry; causes cursor to return to leftmost position on next lowest line. Signals "END OF INPUT LINE."
π		Value of Pi: 3.1415927.

IV. Special keys

The following keys, when pressed while the SHIFT key is being held down, will perform the following functions:

RUN STOP	LOADS and RUNS the next encountered program from the built-in tape unit.
CLR HOME	Clears print from screen and moves cursor to upper left corner of screen. Program statements and all variables are retained.
⇑ CURSOR ⇓	Moves cursor one space up. Will not scroll off top of screen. Does not delete characters as it passes over them.
⇐ CURSOR ⇒	Moves cursor one space left (backspace). Wraps around to rightmost position on next highest line. Does not delete characters as it passes over them.
OFF RVS	Resets reverse field printing to normal printing.
INST DEL	Inserts a space immediately in cursor position. All characters to right of inserted space are moved one space to right. Stops when 80th character is filled.

When the SHIFT key is not pressed, the keys will perform different functions, as indicated:

RUN STOP	Stops execution of command in progress (LIST, LOAD, RUN, etc.).
CLR HOME	Returns cursor to upper left corner of screen.
⇑ CURSOR ⇓	Moves cursor one space down. When cursor is at bottom of screen, print will scroll off top of screen. Does not delete characters as it passes over them.
⇐ CURSOR ⇒	Moves cursor one space right. Will wrap around to left most position of next lowest line. Does not delete characters as it passes over them.

OFF RVS	Enables reverse field print (black characters on a white background).
INST DEL	Deletes character immediately to left of cursor. All characters to right of deletion are moved one space left. Line is filled with trailing blanks if needed.

V. Cleaning your PET

With power switch in "OFF" position, gently wipe keytops with a slightly damp cloth. Do not flood with water.

Use any of the available lens-cleaner sprays to clean the video screen. Spray screen lightly, and dry with a soft, non-linting cloth or tissue.

Wipe the cabinet with a soaped, well-wrung sponge. Do not use any commercial abrasive cleaners. Rinse by wiping with clean, slightly damp sponge or soft cloth. Do not immerse in water.

Clean the recorder unit inside by touching dust particles with slightly damp cloth or sponge. Do not wipe surface . . . cloth will snag on metal or plastic parts and may cause breakage. (See next section for more detailed instructions.)

The outside of the recorder unit may be cleaned in the same manner as the keytops.

VI. Cleaning and demagnetizing your tape deck head

To be performed every 50–100 hours of tape running time or when cassette unit fails to read tapes reliably.

You'll need the following tools and materials:

1) Tape head cleaner. ("NORTRONICS" Brand is recommended.) Do **not** use Tricloroethane or any other plastic or rubber solvent. Alcohol may be used in an emergency, but is not recommended for long term use.

2) Cotton Swabs. "Johnson & Johnson" Brand is recommended; the cotton seems to stick to the end of the swab better.

3) Tape Head Demagnetizer: "NORTRONICS," "HAND-DE-MAG" and "ROBINS" brands are recommended. Unit must have protective plastic or rubber covering on pole piece so as not to scratch delicate head gap.

HOW TO PROCEED:

1) Turn Off PET.

2) Press EJECT to open cover, then press PLAY on tape deck to make heads available.

3) Use tape head cleaner and one side of a cotton swab to clean surfaces of RECORD/PLAY (R/P) and erase head. (See Figure 7.)

   Scrub gently, noting if there is any build-up of tape oxide particles on or around head gap of the R/P head. If so, this is sufficient reason for unreliable performance.

   Also clean pinch roller and other tape bearing surfaces if tape head cleaner is suitable for this purpose. (Check label.)

4) Plug in demagnetizer and activate it while it is at least one foot away from cassette heads.

5) Slowly move demagnetizer up to R/P head and around on head surface. Rate of motion should be approximately one inch per second during this time.

6) Slowly move demagnetizer to erase head and then to all other ferrous metal surfaces which come into proximity with the tape.

7) Now slowly move demagnetizer away from heads and do not de-activate field until demagnetizer is at least two feet away from heads.

Tape head cleaning and demagnetizer procedure is now complete. Inspect R/P nead surface for wear. If tape has worn a groove on head surface more than a couple of tape thicknesses deep and program reading performance is still poor, then replacement of tape head is indicated. (This usually occurs after three thousand or more hours of tape running time.)

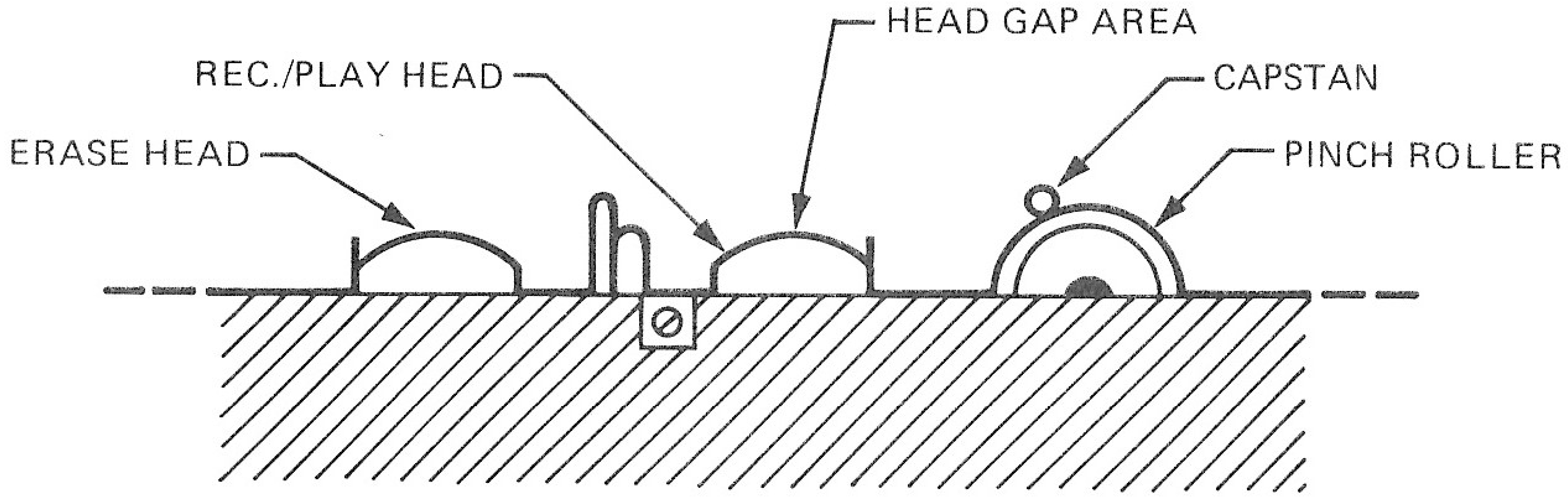


Figure 7. Cassette deck head area

VII. Hints if you have a problem

Every once in a while your PET may seem to be unresponsive or out of sorts or just plain broken.

Before contacting your Dealer or Commodore directly, please follow the instructions.

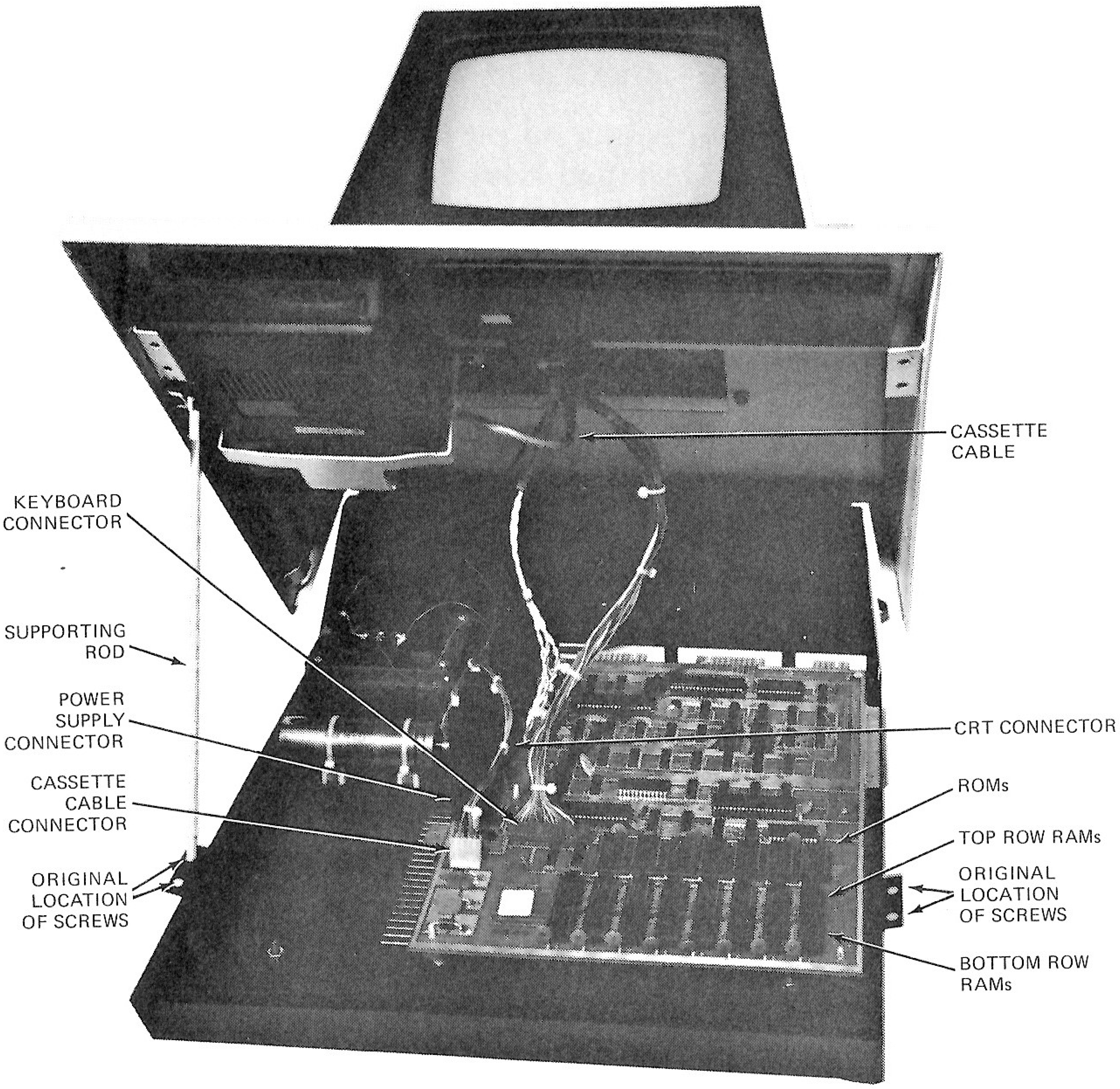


Figure 8. Internal view of PET computer

1. Press the rocker switch to the "Off" position (White Dot does not show.)
2. Remove the power cord from the wall socket to avoid possible electrical shock.
3. Remove the two screws located on each side of the unit under the lip of the cover.
4. Lift the cover slowly – a few inches. When you locate the cable leading to the cassette, remove the connector at the main board. Then lift the cover all the way up and engage the supporting rod located on the left side of the cover.

5. The first two rows of rectangle devices are RAMs. Press gently but firmly on each RAM to be sure each is firmly seated. The third row is a single row of ROMs. Make sure they are firmly in place.
6. The keyboard connector is located in the same row as the ROMs and to their left. Be sure the keyboard cable is firmly connected to it.
7. Left and forward of the keyboard connector is the power supply connector. The power supply cable should be firmly attached to the connector.
8. The CRT connector is behind the keyboard connector and beside the second U-shaped Heat Sink. Be sure the CRT cable is properly connected.

If you're sure everything is firmly in place, disengage the support rod and lower the cover. Before closing it, re-connect the cassette recorder The blue wire on the connector should be closest to you, and the connector will slip on easily.

Close the unit, plug it in and turn the unit on. If you have an 8K unit and the screen shows less than 7167 bytes free (or less than 3071 bytes free if you have a 4K unit), you may have a faulty RAM.

Turn off the unit and unplug the cord. Open the PET, being sure to disconnect the cassette cable.

Following this diagram, locate the suspect RAMS.

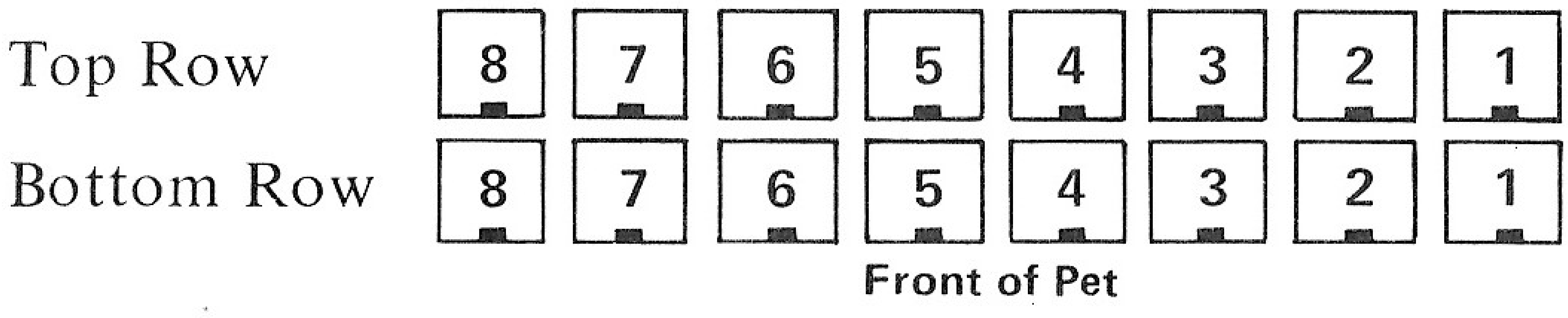


Figure 9. RAM chips on 8K unit

If bytes free are less than 1023, the suspect RAMs are in column 2. The table below tells you exactly where to look, and which column of RAMs to exchange.

If bytes free are less than

	Column
1023	2
2047	3
3071	4
4095	5
5119	6
6143	7
7167	8

If Column 1 RAMs were malfunctioning, your PET would not respond at all.

Having located the suspect RAMs carefully lift both RAMs out of their sockets using a nail file, letter opener or other thin blade, *straight up* or the pins will bend – and then they may not fit back into the sockets.

Be sure to lift the RAM straight up from the socket. Notice the notches at the *front* of the RAM. Be sure pins do not bend. Exchange the top RAM with the bottom RAM. Be sure the pins do not bend or fold. Be sure the notches are in the FRONT of the RAM.

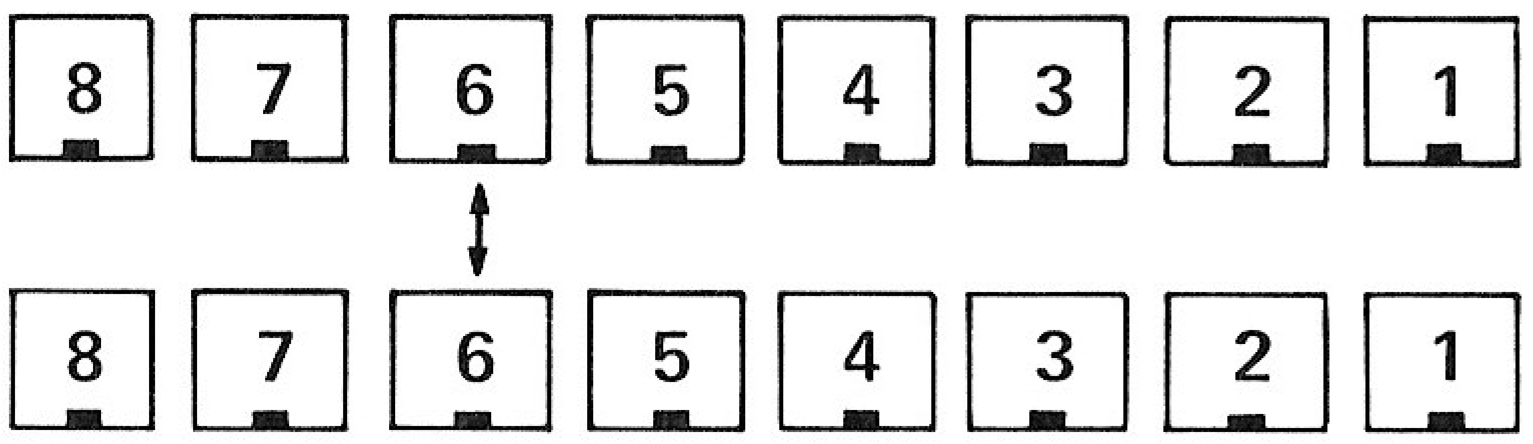


Figure 10. Bytes free are less than 5119 in this example of RAM exchange

Without lowering the cover, plug in the PET and turn it on. If you've exchanged the correct RAMs, the number of bytes free should change. If the number INCREASES, the bad RAM is the one you just placed in the top row. If the number of bytes DECREASES, the bad RAM is the one you just placed in the bottom row. If the number doesn't change, you selected the wrong column and should start over.

Please note that if you have NEVER had experience in removing RAMs from their sockets, you should practice on something *else* before you try exchanging RAMs in the PET. Please remember that bent or broken pins will keep your PET from functioning properly.

When you locate the appropriate RAMs, mail the defective component to:

Commodore PET System Service
901 California Avenue
Palo Alto, California 94304
(415) 326-4000

A replacement will be sent right away. You can use the RAMs in Column 8 to fill the gap and use only 6143 bytes of RAM until the new RAMs arrive.

If the RETURN key isn't doing its job, then you may have a problem with Column 1 RAMs. (For instance, the RETURN key moves to the end of the entry line instead of to the left edge of the screen and one line down – or you press RETURN and the whole screen fills with "garbage," and miscellaneous characters). Follow the RAM exchange procedure – but exchange the top row Column 8 RAM with the bottom row Column 1 RAM, and exchange the top row Column 1 RAM with the bottom row Column 8 RAM. While you'll still have to get a replacement RAM or two, at least PET will respond to your touch. Now if you don't have enough bytes free, you know which RAMs are bad.

If the screen gets the jitters, the chances are your PET is overheating. If your cassette unit doesn't co-operate, refer to the cassette service on page 33, and if you still can't save your program, call us. If the keyboard in general doesn't respond, the wiring may be loose.

There are other explanations, of course, for PET's non- or misbehavior. At that point you may want to call Commodore or the dealer from whom you bought your PET.

VIII. References

Entering BASIC, J. Sack and J. Meadows, Science Research Associates, 1973.

BASIC: A Computer Programming Language, C. Pegels, Holden-Day, Inc., 1973.

BASIC Programming, J. Kemeny and T. Kurtz, Peoples Computer Co., 1010 Doyle (P.O. Box 310), Menlo Park, CA 94025, 1967,

BASIC, Albrecht, Finkle and Brown, Peoples Computer Co., 1010 Doyle (P.O. Box 310), Menlo Park, CA 94025, 1973.

A Guided Tour of Computer Programming in BASIC, T. Dwyer, Houghton Mifflin Co., 1973.

Programming Time Shared Computer in BASIC, Eugene H. Barnett, Wiley-Interscience, L/C 72-175789 ($12.00).

Programming Language #2, Digital Equipment Corp., Maynard, MA 01754.

101 BASIC Computer Games, Software Distribution Center, Digital Equipment Corp., Maynard, MA 01754 ($7.50).

What to Do After You Hit Return, Peoples Computer Co., 1010 Doyle (P.O. Box 310), Menlo Park, CA 94025 ($6.95).

Basic BASIC, James S. Coan, Hayden Book Co., Rochelle Park, N.J.

Advanced BASIC, James S. Coan, Hayden Book Co., Rochelle Park, N.J.

Notes

Notes

Notes

Notes

COMMODORE SALES AND SERVICE:

Commodore Business Machines, Inc.
901 California Avenue
Palo Alto, California 94304, USA

MOS Technology
950 Rittenhouse Road
Norristown, Pennsylvania 19401, USA

Commodore Business Machines, Limited
3370 Pharmacy Avenue
Agincourt, Ontario, Canada M1W2K4

Commodore Business Machines Limited
446 Bath Road
Slough SL1 6BB, England

Commodore Büromaschinen GmbH
Frankfurter Strasse 171-175
6078 Neu Isenburg
W. Germany

Commodore Switzerland S.A.
Bahnhofstrasse 29-31, 2 Stock
Postfach 666, 5001 Aarau, Switzerland

Commodore Japan Limited
Taisei-Denshi Building
8-14 Ikue 1-Chome
Asahi-Ku, Osaka 535, Japan

Commodore Electronics (Hong Kong) Ltd.
Watsons Estates
Block C, 11th floor
Hong Kong, Hong Kong