Chapter 1

GENERAL INFORMATION

A program is a series of instructions which control the operations of a computer. Such instructions must be a part of a predefined set which the computer is designed to understand, and which are combined in accordance with a fixed set of rules. This set of instructions is referred to as the computer's language. The individual instructions (words) used by the language are referred to as commands, statements, or functions, and the rules which govern the manner in which instructions are combined are referred to as the language's syntax.

The programming language supplied with the PX-8 is BASIC (Beginner's All-purpose Symbolic Instruction Code). BASIC for the EPSON PX-8 is an EPSON-enhanced version of Microsoft BASIC which has been expanded by EPSON for use with the PX-8 and which operates under the CP/M operating system of the PX-8 (see the PX-8 User's Manual for further information on the CP/M operating system). BASIC is loaded into the computer's memory from a ROM capsule, by executing the "BASIC" program under CP/M.

Among the enhancements which have been made are:

(i) the addition of a powerful screen editor which vastly increases the ease with which programs are entered, modified and executed

(ii) a variety of graphic statements and functions which take advantage of the PX-8's large 480 by 64 dot display

(iii) statements for selecting different modes of screen operation, such as graphics and split screen

(iv) statements and functions which support communication through the PX-8's RS-232C serial interface

(v) statements which make it possible to use the PX-8's built-in microcassette drive in the same manner as a disk drive

(vi) and statements which make it possible to control the computer's power supply under program instruction.

Other features of this BASIC are as follows.

- BASIC can be made resident in RAM after loading it from ROM capsule, allowing it to be started up almost instantly whenever the PX-8's power is turned on.

- The BASIC program area in memory is divided into five parts, allowing up to five different programs to be held simultaneously. This facilitates development of programs and makes it possible to use multiple program applications.

- The LCD screen can be switched between any of four different screen modes, which are (1) a full screen text mode in which the screen consists of 7 or 8 lines (depending on whether function key definitions are displayed) of 80 columns each; (2) a split screen mode, in which the screen is divided into two consecutive halves of 39 characters each; (3) a twin screen mode, in which two separate areas in display memory are displayed simultaneously; and (4) a graphic mode which is used for drawing figures and diagrams with PX-8 BASIC's graphic statements.

- The microcassette drive built into the PX-8 is supported as a disk device. This means it can be used in almost exactly the same manner as if it were a disk drive. This also applies to auxiliary storage devices such as ROM capsules (which because they are ROMs can only be read from and not written to), and an area in random access memory which is referred to as RAM disk. Floppy disk drives connected to the PX-8's high speed serial port are treated as normal disk drives. The ability to use the PX-8's random access memory in the same manner as if it were a disk drive (RAM disk) is particularly useful because it allows utilization of disk-based functions which ordinarily would require a disk drive. The microcassette drive has some limitations. The main differences are the speed of access and the fact that tape access is sequential, so that random access file handling is not possible.

Procedures for installing and starting up BASIC are described below. Before following these procedures, you may wish to use the CONFIG command of CP/M to select the printer output port (high speed serial or RS-232C) and to specify the default parameters for communication through the RS-232C interface. See the PX-8 User's Manual for details.

1.1 Installing BASIC

BASIC for the PX-8 is distributed in the form of a ROM capsule which must be installed in the back of the PX-8 before BASIC can be loaded. The location of this socket and procedures for changing ROM capsules are as described in the PX-8 User's Manual. BASIC can be inserted in either ROM socket 1 or ROM socket 2 and would be loaded from the ROM drive which has been allocated to that socket as if it were a program on a conventional floppy disk.

If you are using an applications program ROM as regularly as BASIC, and wish to have the applications ROM as well as BASIC, you may consider using PIP to transfer some of the CP/M utility programs from the utility ROM into the RAM disk area or even onto cassette tape. Details of how to use PIP to do this are given in the User's Manual.

NOTES:

1. *BASIC cannot be started if the RAM disk size is set to 24K with the CONFIG command.*
2. *If a ROM capsule is changed for any reason while in BASIC, execute the RESET command.*

1.2 Starting BASIC

When you switch on the PX-8, there are a number of possible states in which the computer can be. If it is in the middle of an applications program (either because the power has been switched off with [CTRL] held down, or the power has automatically switched off because there was no input), then it is necessary to exit from the program before loading BASIC. The other possiblities are that the CP/M command line is displayed, or the MENU screen has been set.

(a) USING BASIC FROM THE CP/M COMMAND LINE

Entering BASIC from the CP/M command line is achieved by treating the BASIC ROM as a program on a disk drive. Thus if the system prompt says "A>", you would need to type "B:BASIC" or "C:BASIC" followed by the [RETURN] key in each case, depending on which socket the ROM has been placed in, and which ROM has been allocated to which drive. (Allocation of drive names is carried out with the CONFIG program which is described in the User's Manual. If you wish to find out in which drive BASIC is located, use the DIR command of CP/M rather than using the CONFIG program). On pressing [RETURN], BASIC will be loaded into RAM. If you wish to run a BASIC program directly you can do so by adding the name of the program and its drive location, as for example running the program "NAME" which is located in drive A: when BASIC is in drive C:

A>C:BASIC A:NAME.BAS

In this case there MUST be a space between the word BASIC and the drive name in which the program sits. The extension ".BAS" showing that "NAME" is a BASIC program is not necessary. However, if a BASIC program has been named with a different extension (e.g. ".GPH" so that all graphics programs are identifiable), then this extension MUST be used; otherwise the computer cannot find the program. Also in this particular case, since the default drive name is A: it is not necessary to type it in before the name of the BASIC program.

(b) ENTERING BASIC FROM THE MENU

The PX-8 has a mode which allows easy loading of programs which are set up on a menu. A description of how to use the MENU is given in the User's Manual, and details of setting up the menu for use with BASIC is given in section 1.4.2 (Warm starts) of this manual.

If you have set up the MENU so that BASIC is one of the files to appear on

it there are still a number of possibilities, depending on whether BASIC has been used previously and on whether you wish to run a BASIC program directly. The possibilities are as follows:

(i) BASIC.COM is the first file on the MENU.
(ii) BASIC.COM is on the MENU but not the first file.
(iii) BASIC is resident in memory.
(iv) You wish to RUN a stored BASIC program.
(v) You wish to run a BASIC program directly when BASIC is resident.

(i) BASIC.COM is the first file on the MENU

The simplest case is when BASIC has not been used when the PX-8 is switched on. If the MENU has been set up to show the files on the drives allocated to the ROM sockets, the appearance of the screen will be as follows if BASIC.COM is the first file in the main MENU area in the top left hand corner. The command line will have "BASIC" entered by the computer, together with the drive name prefix. BASIC.COM with its drive prefix will be flashing in the main menu area.

```
*** MENU screen ***  01/01/84 (SUN) 10:00:23   54.5k CP/M  ver 2.2 PAGE 1/1
B:BASIC
B:BASIC     COM
```

Simply pressing the [RETURN] key will load BASIC into memory from the ROM. BASIC will take a few seconds to load. Section 1.3 describes what to do next.

WARNING:
In loading BASIC into memory, any other programs already there will be destroyed.

(ii) BASIC.COM is not the first file on the MENU

Depending on how the MENU was set up, BASIC.COM may not be the first file on the MENU. For example

```
*** MENU screen ***  01/01/84 (SUN) 10:10:37   54.5k CP/M  ver 2.2 PAGE 1/1
            A:GRAPH.BAS
A:GRAPH     BAS         B:BASIC     COM
```

The MENU screen will automatically put the first program in the top left hand position onto the command line. If, as in the screen above, the first program is not BASIC.COM, then you must put BASIC onto the command line using the cursor keys to select it. The screen display will then change as follows

```
*** MENU screen ***  01/01/84 (SUN) 10:11:11   54.5k CP/M  ver 2.2 PAGE 1/1
B:BASIC
A:GRAPH    BAS       B:BASIC    COM
```

and BASIC.COM will be flashing in the main MENU area. Now BASIC can be loaded by pressing the RETURN key, as in the previous example.

(iii) BASIC is resident in memory

If BASIC had been used when the PX-8 was switched off, instead of the MENU screen showing BASIC or indeed another program on the command line, the following display would come up

```
*** MENU screen ***  01/01/84 (SUN) 10:12:07   54.5k CP/M  ver 2.2 PAGE 1/1
BASIC    (resident) A:GRAPH    BAS       B:BASIC    COM
```

and the first program position showing "BASIC (resident)" would be flashing. The command line is empty when this occurs.

Simply pressing RETURN will enter BASIC without loading it.

(iv) Running a BASIC program directly

The MENU can also be used to select and RUN a BASIC program directly. If the program is selected by means of the cursor keys, the screen will appear as follows.

```
*** MENU screen ***  01/01/84 (SUN) 10:26:58   54.5k CP/M  ver 2.2 PAGE 1/1
B:BASIC    A:GRAPH.BAS
BASIC    (resident) B:BASIC    COM       A:GRAPH    BAS       A:SAMP1    BAS
A:SAMP2    BAS
```

Note the appearance of the command line. This means that BASIC will be loaded first, and then the program in order to RUN it.

WARNING:
If a BASIC program is RUN in this way, all the BASIC programs in memory which lie in the BASIC program areas 1-5 WILL BE DESTROYED.

If the MENU is used to run a BASIC program directly when the program is put onto the command line, it should appear as in (iv) above. If it does not, you have made an incorrect entry when setting up the MENU option on the System Display, and should refer to the User's Manual to see the correct way to set up BASIC programs.

(v) Running a BASIC program directly when BASIC is resident

The situation may occur that the program is in memory when BASIC is resident. Whereas it is possible to go to the BASIC program menu and then run the program as described in the next section, it may be more convenient to run a program directly from the MENU screen. This can be achieved using the following commands which are described fully in Chapter 3.

/P:n

where n is a program area from 1 to 5 (e.g. /P:4 means program area 4), will enter BASIC and login to this area.

```
*** MENU screen ***  01/01/84 (SUN) 10:31:33   54.5k CP/M  ver 2.2 PAGE 1/1
/P:4_
BASIC    (resident) B:BASIC    COM       A:GRAPH    BAS       A:SAMP1    BAS
A:SAMP2    BAS
```

/R:n

where n is the program area, will enter BASIC and run the program in the specified program area.

```
*** MENU screen ***  01/01/84 (SUN) 10:31:59   54.5k CP/M  ver 2.2 PAGE 1/1
/R:2_
BASIC    (resident) B:BASIC    COM       A:GRAPH    BAS       A:SAMP1    BAS
A:SAMP2    BAS
```

1.3 The BASIC Program Menu

Once BASIC has been started, the BASIC program menu is displayed as shown below.

```
EPSON BASIC ver-1.0 (C) 1977-1983 by Microsoft and EPSON
Move cursor, RETURN to run or SPACE to login.
      ■ P1:              0 Bytes
        P2:              0 Bytes
        P3:              0 Bytes
        P4:              0 Bytes
        P5:              0 Bytes
           14749 Bytes Free
```

This might look slightly different if BASIC was already resident, in which case a display such as the following could appear.

```
EPSON BASIC ver-1.0 (C) 1977-1983 by Microsoft and EPSON
Move cursor, RETURN to run or SPACE to login.
      ■ P1:GRAPH        27 Bytes
        P2:              0 Bytes
        P3:              0 Bytes
        P4:              0 Bytes
        P5:              0 Bytes
           14722 Bytes Free
```

Again, if the command line was of the form shown in (iv) of section 1.2 or the entry from the CP/M command line was to run a BASIC program directly, the above menu would only flash up briefly before the program began running.

The BASIC program menu shows the number of bytes of program text contained in each of BASIC's five program areas (P1 to P5) and the number of free bytes of memory which are available for use as string area, variables, or additional program text. (The BASIC interpreter automatically handles allocation of memory between the three of these as appropriate.)

The following keys can be used while the BASIC program menu is displayed.

[↑]	Moves the cursor upward.
[↓]	Moves the cursor downward.
[RETURN]	Logs in (selects) the program area indicated by the cursor and executes the program which is present in that area.

[SPACE]	Logs in the program area indicated by the cursor, but does not execute the program in that area.

Since the cursor is shown to the left of "P1:" at this point, pressing the space bar here logs in program area 1. To select another program area, move the cursor up or down with the "up" and "down" arrow keys before pressing the space bar.

If no program exists in any area (shown by the length of program text being 0 bytes), then [RETURN] simply logs into that area. On logging into such an area the screen appears thus

```
EPSON BASIC ver-1.0 (C) 1977-1983 by Microsoft and EPSON
14749 Bytes Free
P1:              0 Bytes
Ok
■
```

The "Ok" displayed at the end of this message indicates that BASIC is at the command level; in other words, that it is ready to accept commands. At this point the BASIC interpreter may be used in either of two modes: direct (immediate) mode or indirect (execution) mode.

Commands and statements entered in the direct mode are not preceded by program line numbers. Instead, they are executed immediately when the [RETURN] key is pressed. Since commands/statements entered in this mode are executed as they are input, the results of arithmetic and logical operations are displayed immediately (they can also be assigned to variables for later use). However, the commands themselves are lost once they have disappeared from the screen. This mode is useful for debugging and for using BASIC for simple, non-repetitive arithmetic operations.

The indirect mode is the mode which is used for entering programs. In this mode, commands, statements, and functions are preceded by line numbers which indicate the order in which they are to be executed. However, commands and statements entered in the indirect mode are stored in memory without being executed; execution is deferred until the program is RUN. The indirect mode is used when working with complicated calculations or operations which must be performed many times or stored to be recalled for use at a later date.

From the point of view of the user, the only difference between the direct mode and the indirect mode is that commands and statements entered in the indirect mode must be preceded by line numbers. The computer automatically switches from one mode to the other according to whether commands/statements are preceded by line numbers.

1.4 Warm Starts and Cold Starts

There are two methods for starting up BASIC: the cold start and the warm start.

1.4.1 Cold starts

A cold start is made when the BASIC interpreter is loaded into the memory of the PX-8 from ROM capsule. This type of start is made by executing the BASIC command or by selecting the BASIC.COM file from the menu.

NOTE:
The BASIC program areas are cleared whenever a cold start is made. A cold start must be made whenever the PX-8's power switch is turned on unless it is made resident in the PX-8's memory (the area in which utility programs such as the BASIC interpreter are stored while they are being executed). In this case, BASIC can be started up by making a warm start.

1.4.2 Warm starts

A warm start is the procedure by which BASIC is restarted when it is already present in memory. BASIC becomes resident in memory when loaded, and remains there when the power is turned off if the MENU screen function has been turned on. In this situation, the screen appears as shown below when the power switch is turned back on.

```
*** MENU screen ***  01/01/84 (SUN) 10:40:41   54.5k CP/M  ver 2.2 PAGE 1/1
BASIC    (resident) B:BASIC    COM
```

Procedures for turning on the MENU screen function are as follows.

(1) Turn on the computer's power switch and if the system prompt ("A>" or another drive name) is displayed, proceed as follows:

(2) Press the CTRL and HELP (SYSTEM) keys together; this causes the System Display to appear as shown below.

```
*** SYSTEM DISPLAY ***    01/01/84 (SUN) 10:41:35

<RAM   DISK> 009   kb     <AUTO START>
<USER  BIOS> 000 256 b    <MCT   MODE>    stop,  nonverify <COUNT>   65535
<MENU DRIVE> CBA          <MENU  FILE> 1 .COM    2 .       3 .       4 .
- Select number or ESC to exit.
  1=password  2=alarm/wake  3=auto start  4=menu  5=MCT
   <<  /          <  /mount          /dirinit     >> /erase       /
```

(3) Now press the [4] key to select the menu specification mode; this causes the screen to change as follows.

```
*** SYSTEM DISPLAY ***    01/01/84 (SUN) 10:42:04

<RAM   DISK> 009   kb     <AUTO START>
<USER  BIOS> 000 256 b    <MCT   MODE>    stop,  nonverify <COUNT>   65535
<MENU DRIVE> CBA          <MENU  FILE> 1 .COM    2 .       3 .       4 .
- Select number or ESC to return.
  <MENU>  1=off  2=on  3=drive  4=ext1  5=ext2  6=ext3  7=ext4
```

(4) When the screen changes as shown above, press the [2] key; the display will also show "<MENU>" towards the top right of the screen to show that the menu option is set.

```
*** SYSTEM DISPLAY ***    01/01/84 (SUN) 10:44:07          <MENU>

<RAM   DISK> 009   kb     <AUTO START>
<USER  BIOS> 000 256 b    <MCT   MODE>    stop,  nonverify <COUNT>   65535
<MENU DRIVE> CBA          <MENU  FILE> 1 .COM    2 .       3 .       4 .
- Select number or ESC to return.
  <MENU>  1=off  2=on  3=drive  4=ext1  5=ext2  6=ext3  7=ext4
```

This causes BASIC to become resident in memory the next time it is loaded, as well as causing the PX-8's MENU screen to be displayed the next time BASIC operation is ended. Finish by pressing the [ESC] key once to return to the System Display, then once again to redisplay the system prompt ("A>").

(5) Pressing [CTRL] + [C] or the [STOP] key at this point causes the system to go immediately to the MENU screen, where BASIC can be loaded and executed simply by pressing the [RETURN] key. (BASIC can also be loaded and executed at this point by entering "B:BASIC" or "C:BASIC", depending on the ROM socket into which BASIC has been inserted as described in section 1.1 and which drive it has been allocated to as described in section 1.2(a). Log into one of the BASIC program areas by pressing the space bar, then return to the system by entering SYSTEM and pressing the [RETURN] key.) The MENU screen will now appear as shown below.

```
*** MENU screen ***  01/01/84 (SUN) 10:43:12   54.5k CP/M  ver 2.2 PAGE 1/1

BASIC    (resident) B:BASIC    COM
```

The words "BASIC (resident)" at the left side of the screen indicate that BASIC is resident in memory. At this point, these words should be flashing on and off; this indicates that BASIC execution will be restarted when the [RETURN] key is pressed.

BASIC will remain resident in memory until one of the following actions is performed:

(1) One of the system utilities is executed;
(2) A cold start of the system is made (see the PX-8 User's Manual);
(3) The MENU screen function is turned off from the System Display (see the PX-8 User's Manual);
(4) The MENU screen is terminated by pressing the [ESC] key.

Once BASIC has been made resident, programs in the BASIC program areas will be retained even if the PX-8's power switch is turned off and back on again. Further, the maximum number of files, upper memory limit, and so forth are maintained from the last time BASIC operation was ended.

WARNING:
If the MENU screen function is switched off via the System Display, BASIC will not be resident and the programs in the BASIC program areas will be lost. It is always good practice to save all programs as well as leaving them in memory.

1.5 Ending BASIC Operation

BASIC operation is ended and control returned to the CP/M operating system (or the MENU screen) by typing SYSTEM and pressing the [RETURN] key. (This command can also be executed from within a program.)

Example

```
10 ...
20 ...
30 ...
SYSTEM    RETURN
A>
```

Note:
In the example above, the symbol "[RETURN]" indicates that the operator hits the [RETURN] key. This also applies to other examples throughout this manual.

BASIC operation is also terminated when the PX-8's power goes off. This occurs under the following circumstances.

(1) When the PX-8's power switch is turned off;

(2) When there are no entries typed in from the keyboard for the amount of time specified with the POWER <duration> command while the PX-8 is standing by for input (either at the command level or during execution of an INPUT statement);

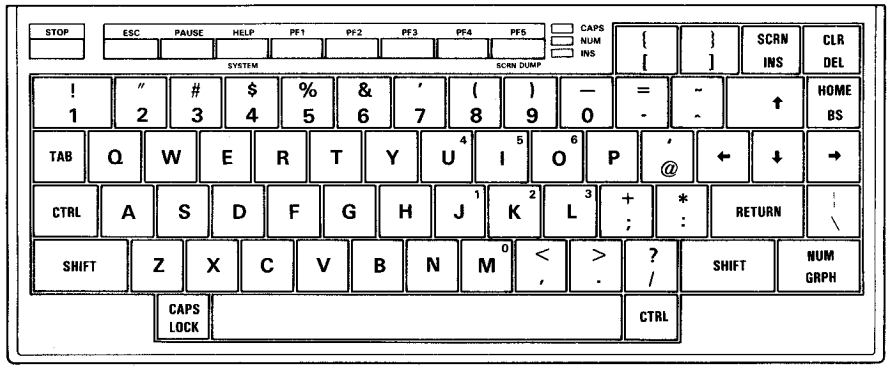
(3) When the POWER OFF command is executed.

If the POWER OFF command has been executed, when the PX-8 is turned on again, the MENU screen will be the resumption point. It is possible to have BASIC operating as it was at the point at which the power was switched off if one of the following conditions is met:

(1) The power switch is turned off while pressing the [CTRL] key;
(2) The power switch is turned off during execution of a command or program; or
(3) The power goes off because no entries have been typed in from the keyboard while the PX-8 is standing by for input.

1.6 Functions of Special Keys in the BASIC Mode

The keyboard of the PX-8 includes a number of special keys as indicated in the figure below.



Functions of these special keys during BASIC operation are as follows.

[PF1] to [PF5] The Programmable Function Keys

These are the PX-8's programmable function keys. Each key contains two functions, the second of which is accessed by pressing the shift key as well as the function key. In this case the keys are numbered [PF6] to [PF10], for instance, [PF4] when shifted is known as [PF9]. Any string of up to 15 characters can be assigned to each of these keys with the KEY command of BASIC or the CONFIG command of CP/M; afterwards, that string of characters can be entered simply by pressing the applicable programmable function key. This can greatly reduce the need to type in commands one character at a time from the keyboard. See the explanation of the KEY command in Chapter 4 and the discussion of the CONFIG command in the PX-8 User's Manual.

There is a third function of each key which is accessed by pressing the [CTRL] key at the same time as the programmable function key. This causes a machine code subroutine to be executed. In the case of the [PF5] key this has been set to dump the screen to a printer. The others can be set by the user. Procedures for doing this are described in the PX-8 OS Reference Manual.

When BASIC operation is started, the first seven characters of the character string assigned to each of the programmable function keys is displayed at the bottom of the screen. Definitions for the unshifted and shifted functions are displayed together (separated by a slash). Display of these definitions can be turned on or off as required with the SCREEN command (see Chapter 4) or by using a control sequence as follows:

```
PRINT CHR$(27); CHR$(&HD3); CHR$(1)   will turn the display off
PRINT CHR$(27); CHR$(&HD3); CHR$(0)   will turn the display on
                                      again.
```

The default settings of the function keys are as follows:

PF1	auto	PF6	(shifted PF1) load"
PF2	list	PF7	(shifted PF2) save"
PF3	edit	PF8	(shifted PF3) system
PF4	stat	PF9	(shifted PF4) menu^M
PF5	run ^M	PF10	(shifted PF5) login

With function keys such as PF5 where there is a carriage return in the command, the command will be executed as soon as the key is pressed. Some (such as PF8) have been defined without a carriage return to ensure that the user makes the final decision on their execution, since if a mistake were made the results could destroy a program. Others (such as PF6) must have further input or an error message will be generated. There are also commands such as LIST which can have optional characters added. Thus typing the letter "L" followed by PF1, will result in Llist being generated, and a listing will be printed on an external printer when RETURN is pressed. Similarly, pressing PF1 and then the characters " – 100" will generate LIST-100 and the lines of the program in the current logged in area will be listed to the screen as far as line 100.

If you are typing many lines of a program using a particular command frequently, it may be useful to change one of the function keys temporarily using the KEY function of BASIC. However, the value you use will be lost when BASIC is cold started and reset to the default values.

PAUSE

The PAUSE key makes it possible to temporarily suspend listing of a BASIC program with the LIST command, or to temporarily stop execution of a BASIC program. (The same result is obtained by pressing the CTRL and S keys

together). The PAUSE condition is then released by pressing any key other than STOP or CTRL and C . Pressing either of these will terminate either the listing or the program execution.

SCRN DUMP (CTRL + PF5)

The SCRN DUMP key outputs the contents of the screen to the printer. The screen contents are output to the printer as ASCII character codes when screen modes 0, 1, or 2 (the text modes) are selected, and in bit image format when the screen mode 3 (the graphic mode) is selected. The same result is obtained by executing the COPY statement in BASIC. (A detailed explanation of the screen modes are given in Chapter 2).

STOP

This key interrupts execution of BASIC programs and returns the BASIC interpreter to the command mode. (The same result is obtained by pressing the CTRL and C keys together). Execution of the interrupted program can then be continued by entering the CONT command. This key is also used to terminate automatic program line number generation initiated by the AUTO command.

The STOP KEY command can be used to disable or re-enable the STOP key (see the section on STOP KEY in Chapter 4).

CTRL + STOP

Pressing CTRL and STOP together during write or read access to the microcassette drive forcibly terminates the access operation and generates an error. (This function cannot be disabled by the STOP KEY command in BASIC). However, if a search is being performed (if the tape is being wound to a specific counter reading to prepare for access) CTRL + STOP will not stop the drive until the search is completed.

CTRL + HELP

Pressing the CTRL and HELP keys together switches operation back to the System Display mode. This means that options such as setting the MENU screen, and setting the alarm and wake independently of BASIC can be carried out just as at the CP/M command level.

Chapter 2

PROGRAMMING CONCEPTS

This chapter discusses a variety of concepts which are applicable to programming in BASIC for the PX-8. These are presented in logical order, with the most fundamental concepts presented first. Mastery of the information included in this chapter is essential to realizing the full potential of the capabilities of BASIC.

2.1 Program Lines and Statements

All BASIC programs are composed of one or more lines, each of which begins with a line number, ends with a carriage return (or RETURN), and includes one or more commands, statements, or functions. The line numbers indicate the order in which the lines are stored in memory and executed. Lines numbers are referenced when the program is edited or when the flow of execution is switched from one point in a program to another. (See the descriptions of the GOTO and GOSUB statements in Chapter 4). Line numbers must be within the range 0 to 65529.

Commands and statements are words in the BASIC language which instruct the computer to perform specific operations. Each line of a program may consist of a single statement, or several statements which may be included on one line; in the latter case, each statement must be separated from the one following by a colon, and the total length of each program line cannot exceed 255 characters.

2.2 Multiple Program Areas

In PX-8 BASIC the BASIC program area is divided up into five parts, making it possible for up to five separate BASIC programs to be present in memory simultaneously. The program area selected is determined by the /R: or /P: options when BASIC is started. Once BASIC has been started, program areas can be switched by executing the LOGIN command (either in the direct mode or from within a program); this makes it possible to chain execution of programs between areas.

Further, the TITLE command can be used to assign names to the program areas displayed in the BASIC program menu; this command includes an optional parameter which can be specified to prevent the program in the applicable area from being edited or accidentally erased.

In addition the STAT command makes it possible to determine the status of the currently selected program area or other program areas.

The BASIC program areas are managed on a dynamic basis; that is, BASIC allocates memory to each of the areas according to the size of the program that area contains.

2.3 Screen Editor

The screen editor is a feature of EPSON BASIC which makes it easy to enter and edit the lines of BASIC programs. This ability is central to programming the PX-8 in BASIC.

The screen editor uses the concept of logical lines for display of commands and statements (in the direct mode) or program lines (in the indirect mode). A logical line is a collection of characters which is handled by the screen editor as one logical unit, regardless of the number of physical screen lines which it may occupy. Normally, a logical line is terminated by pressing the [RETURN] key.

During typing, logical lines are automatically continued when the cursor moves from the right side of the screen to the beginning of the following physical line. This applies regardless of whether the cursor is moved by typing characters or spaces, or by the [TAB] key.

There are several methods of editing lines of BASIC programs which have previously been stored in memory. The most primitive is simply to retype the entire line using the same line number. The BASIC interpreter automatically replaces the old line with the new one when the [RETURN] key is pressed.

However, the screen editor makes it possible to edit a logical line (after displaying it, if necessary, with the EDIT or LIST commands; these commands are explained in detail in Chapter 4) by moving the cursor to that line with the cursor controls, then making changes using the screen editor's control keys. A variety of keys are provided for use with the screen editor; these keys are described below.

[↑] [→] [↓] [←]

At the command level, these keys move the cursor (the flashing square which appears on the screen during BASIC operation) in the directions indicated by the arrows on the key tops. These keys are equipped with a repeat function which moves the cursor continuously at a steady rate when any of the keys is held down.

[TAB]

The [TAB] key moves the cursor to the right from its current location to the next tab position on the screen. The liquid crystal display screen of the PX-8 has 10 tab positions, starting with the column on the far left side of the screen. (A column consists of one character width in the same position on each line of

the screen). Subsequent tab positions are located in every eighth column. The same effect is achieved by pressing the CTRL and I keys together.

BS

The BS (backspace) key deletes the character located immediately to the left of the cursor and moves the remainder of that logical line to the left by one character position. This key does nothing if pressed while the cursor is at the beginning of a logical line. The same effect is achieved by pressing the CTRL and H keys together.

CTRL + A

Pressing these keys together moves the cursor to the beginning of the logical line in which it is currently located.

CTRL + B

Pressing these keys together moves the cursor to the first character of the word preceding its current position. For the purpose of this function, a word is any group of letters which is separated from other letters by a space or special character. The same result is achieved by pressing the SHIFT and ← keys together.

CTRL + F

Pressing these keys together moves the cursor to the first character of the word following its current position. The same result is achieved by pressing the SHIFT and → keys together.

CTRL + J

Pressing these keys divides the logical line into two parts at the current position of the cursor. When the cursor is already located at the beginning of a logical line, it inserts a logical line consisting only of spaces. If the cursor is positioned to the right of the last character in a logical line, it inserts a logical line consisting entirely of spaces between the current logical line (the line in which the cursor is located) and the following one.

CTRL + X

Pressing these keys together moves the cursor to the position following the last character in the current logical line.

HOME (SHIFT + BS)

Pressing these keys together moves the cursor to the home position without clearing the screen. The same result is achieved by pressing the CTRL and K keys together.

DEL

Pressing the DEL key deletes the character which is located at the position of the cursor and moves the remainder of the logical line to the left by one character position. No characters are deleted if this key is pressed while the cursor is at the end of a logical line.

CTRL + E

Pressing these keys together deletes all characters from the cursor position to the end of that logical line.

CTRL + Z

Pressing these keys together deletes all characters from the cursor position to the end of the screen.

CLR (SHIFT + DEL)

This key clears the entire screen and moves the cursor to the home position (the upper left corner of the screen). The same effect is achieved by pressing CTRL and L

INS

Pressing this key once places the screen editor in the insert mode; pressing it again (or pressing any of the cursor control keys or the RETURN key) restores normal operation. In the insert mode, the cursor and characters from the cursor to the end of the logical line are moved to the right by one position when any character key is pressed; the character typed is then inserted at the cursor's former position. The red INS LED built into the keyboard lights when the screen editor is in the insert mode, and the cursor changes from block form to underline form.

The screen editor can also be placed in the insert mode by pressing CTRL and R together.

RETURN

Pressing this key executes direct commands in the logical line in which the cursor is located or stores program lines in the computer's program text area. Operation is the same no matter where the cursor is located in the logical line. The same effect is achieved by pressing the CTRL and M keys together.

STOP

The principal function of this key is to halt program execution. Pressing this key in the command mode moves the cursor from the logical line in which it is currently positioned to the beginning of the next logical line. This key is also

used to terminate automatic program line number generation by the AUTO command. The same result is achieved by pressing the [CTRL] and [C] keys together.

[CTRL] + [→]
Pressing these keys together switches the cursor to virtual screen 2. This key is effective only while BASIC is in the program input mode, and cannot be used during execution of an INPUT or LINE INPUT statement. (See section 2.13 below for an explanation of the PX-8's virtual screens.)

[CTRL] + [←]
Pressing these keys together switches the cursor to virtual screen 1. This key is effective only while BASIC is in the program input mode, and cannot be used during execution of an INPUT or LINE INPUT statement.

2.4 EDIT Mode

In addition to the screen editor, PX-8 BASIC features an edit mode which increases the efficiency of program editing by making it possible to scroll to any point in a program. The edit mode is entered by executing the EDIT command in the direct mode (see the explanation of the EDIT command in Chapter 4), and is terminated by pressing the [CLR] or [ESC] keys.

The keys used for scrolling, cursor movement and program editing in the edit mode are basically the same as with the normal screen editor. However, functions of certain keys differ as follows:

(1) Cursor control keys
In the edit mode, the cursor control keys ([←] , [→] , [↑] and [↓]) move the cursor in the same manner as when the screen editor is used in the normal direct mode. However, when the cursor is moved to the logical line at the top or bottom of the screen, that line is automatically scrolled as necessary to bring it completely inside the real screen. (This operation is performed when part of the program has been moved beyond the outside of the screen by previous scrolling).

(2) [SHIFT] + [↑]
When only part of a logical line is displayed at the top of the screen, pressing [SHIFT] and [↑] together scrolls the screen so that the entire logical line is displayed, then moves the cursor so that it is positioned at the beginning of that logical line. Otherwise, the screen window is scrolled as necessary to display the logical line preceding that displayed at the top of the screen.

(3) [SHIFT] + [↓]
When only part of a logical line is displayed at the bottom of the screen, pressing [SHIFT] and [↓] together scrolls the screen so that the entire logical line is displayed, then moves the cursor so that it is positioned at the beginning of that logical line. Otherwise, the screen window is scrolled as necessary to display the logical line following that displayed at the bottom of the screen.

(4) [CTRL] + [↑]
Pressing [CTRL] and [↑] together clears the screen, displays the program's first line, and moves the cursor to the beginning of that line.

(5) [CTRL] + [↓]
Pressing [CTRL] and [↓] together clears the screen, displays the program's last line, and moves the cursor to the beginning of that line.

(6) [CLR]
Pressing this key clears the screen and terminates operation in the edit mode.

(7) [ESC]
Pressing the [ESC] key terminates operation in the edit mode without clearing the screen.

When editing programs, remember that changes made on the screen are not reflected in the program in memory until the [RETURN] key has been pressed with the cursor located in that line. This applies both in the edit mode and to changes made using the screen editor in the normal mode.

Direct mode commands can be executed in the edit mode; however, BASIC returns to the normal command level after execution is completed (unless the command executed is the EDIT command).

2.5 Using the Screen Editor and EDIT

The use of the edit and screen editor modes are best seen with the help of an example:

Use an empty program area or clear the memory in the area currently logged in by typing NEW and pressing [RETURN], then type in the following line of a BASIC program:

10 REM This is a remark statement

When you press [RETURN], but not until you do, the line will be stored as a BASIC program line. This line can be edited simply by using the screen editor. Use the cursor keys to move up the screen so that the cursor lies over one of the characters of the line which has just been typed in.

With the [CTRL] key held down, press the [X] key. The cursor now jumps to the end of the line so that you can add more characters to it.

With the [CTRL] key held down, press the [A] key and see that the cursor returns to the beginning of the line. Type a "2" so that the line becomes:

20 REM This is a remark statement

Now holding down the [SHIFT] key, press the right cursor key five times. Each time it is pressed the cursor jumps to the first character of the next word. With five jumps, it will be on the "r" of "remark". Use the shifted left cursor key to move the cursor back in a similar way.

Reposition the cursor at the beginning of "remark" and with the unshifted left cursor key place the cursor in the space before the"r". Now press the [INS] key. The LED next to INS will light to show that the PX-8 is in insert mode, and the cursor will become a flashing underline character. Type a word such as "new " and press the [RETURN] key.

Now if you LIST the program you will find that it consists of the following two lines:

10 REM This is a remark statement
20 REM This is a new remark statement

This means of duplicating lines with the same or similar statements but with different line numbers can be very useful in saving a great deal of typing, especially when a program has a number of similar subroutines or loops.

Sometimes it is necessary to split one line into two lines. For example add line 30 to the program to read as follows:

30 FOR N = 1 TO 20 : NEXT N

Move the cursor onto line 30 again and move the cursor to the colon using the shifted right cursor key. Pressing the DEL key will remove the colon if the cursor is directly on top of it. If the cursor is to the right of the colon it can be removed by using either CTRL + H or the BS key. Now press the INS key and add " 50 " so that the line appears as follows:

30 FOR N = 1 TO 20 50 NEXT N

This is an incorrect BASIC line. Move the cursor to the space before the "5" and press the J key while holding down the CTRL key. The characters from "50" onwards will be moved to the next line but will not be added to the BASIC program. However, if the RETURN key is pressed a line 50 will be added to the program. The program will now have the following lines if listed:

10 REM This is a remark statement
20 REM This is a new remark statement
30 FOR N = 1 TO 20 : NEXT N
50 NEXT N

Note how line 30 has remained unchanged, because in exiting from it the RETURN key was not used. Since in moving the "NEXT N" to line 50, line 30 has been left with a surplus statement, it has to be removed. Move the cursor to line 30 and use CTRL + X to move the cursor to the end of the line, then the shifted left cursor key to move it back to the "N" of "NEXT". The unshifted left cursor key can be used to bring it on to the colon. If the shifted cursor key is used, the cursor will move back too far.

Now press CTRL + E . The line will clear from the cursor onwards and by pressing RETURN the truncated line can be entered, as can be seen by listing the program again.

If you wish to clear the rest of the virtual screen from the cursor onwards CTRL + Z can be used instead.

Add line 40 to the program as follows:

40 PRINT "This is line 40"

Now move the cursor up to the "4" of line 40 and make it into line 60 by overtyping the "4" with a "6". Move along the line using the shifted right cursor key or the alternative CTRL + F . Alter the "4" to a "6" here also.

When line 60 has been added to the program by pressing the RETURN key, add a further line 70 in the same way. It may be faster to use CTRL + X to move to the end of the line then the shifted left cursor key to move the cursor back to the number "6" in the string. Just as CTRL + F can be substituted for the right cursor key, so CTRL + B can be used to move back instead of the left cursor key.

Now add the lines:

80 FOR J = 1 TO 10
90 PRINT J * J
100 NEXT J

Until now only the normal screen editor has been used. In most cases this is all that is required, since a program can be listed and edited by moving around the screen. The important point to remember is that the line is only altered in the stored program IF THE RETURN KEY IS PRESSED.

The EDIT mode has some advantages if a number of lines are to be edited and if the editing is to be carried out on a screen with a limited number of lines, e.g. in screen mode 3.

To illustrate the use of the edit mode, type

SCREEN 3,0,0 : LIST

The screen will clear and show lines of the program from 50 onwards, since the previous ones have scrolled off the virtual screen; in this screen mode it is limited to 8 lines, the same as the window. The cursor will be seen as a non-flashing underline character on the bottom line of the screen. Using LIST would be slow if it was necessary to edit a number of lines in this screen mode. However, EDIT makes a considerable difference.

Type EDIT and press the RETURN key.

The screen will clear and display line 10. This can be edited normally with the screen editor. Since the screen displayed is screen mode 3, moving the cursor to the bottom of the screen with the cursor key will not cause line 10 to scroll off the top of the screen. However, it is possible to scroll through the program by using the shifted ↑ and ↓ cursor keys. The screen editor can be used as described above when a particular line needs to be altered.

In programming, it is often necessary to know which is the first or last line of a program, because a new subroutine needs to be added or a constant inserted at the beginning of the program. If the PX-8 is in EDIT mode, pressing CTRL and the ↑ cursor key will place the first line on the top of a blank screen ready for editing. If the CTRL and ↓ keys are pressed, the last line of the program is displayed instead.

To illustrate the use of these keys, type edit 50, so that line 50 is displayed on the screen. Now press CTRL and the ↑ key. The screen will clear and line 10 will be displayed. Press the SHIFT and ↑ keys. Because there is no lower line number, the cursor will be placed on a blank line, above line 10. Type in a line 5 as follows, remembering to press RETURN to enter it as a line of the BASIC program.

5 A$="THE END"

Now press CTRL and the ↓ key. You can see that the last line of the program is line 100. Whereas the SHIFT and ↓ keys can be used to move to a new line, it is better simply to press RETURN because the contents of line 100 will still be visible. If the CTRL and ↓ keys are used the screen will be clear because line 100 will be scrolled off the top.

Now type in a line 110 to complete the program:

110 PRINT A$

There is one other aspect of using the EDIT mode, which applies to any screen mode, which makes it preferential to using LIST and the screen editor. This may be illustrated as follows:

Edit line 10 to read as shown so that it runs over more than one line of the screen:

10 REM This is a remark statement which has now been extended to run over on to the next line of the screen

List the program to line 60. Note that the screen shows only half of line 10. Move the cursor up to this line and try altering a character. When you press RETURN, a "syntax error" message will be shown because you tried to enter a line which was not logically correct for BASIC. The screen couldn't "see" the first part of the line so it rejected the line as nonsense.

Clear the screen and list line 10. With the full line present on the screen move the cursor to the second part of the line and make an alteration. Pressing the RETURN key will cause the change to be accepted. The EDIT mode only allows complete lines to appear on the screen, and so prevents the syntax error obtained using list.

To illustrate this type EDIT and when line 5 has been displayed use the SHIFT and ↓ keys to display successive lines of the program until line 80 is reached. At this point, only the part of line 10 which overruns will be displayed on the screen. Now move the cursor to the top of the screen. As soon as the cursor reaches the top line, the whole of line 10 is displayed. This would not happen in the normal screen editor mode.

2.6 Types of Data

2.6.1 Text data — The ASCII character set

The ASCII character set is a set of characters which are internally represented in the form of 1-byte† numeric codes and converted to characters for display by the PX-8's character generator.

The character generator of the PX-8 includes character sets for the following nine countries.

1. Denmark
2. England
3. France
4. Germany
5. Italy
6. Norway
7. Spain
8. Sweden
9. United States

Any of these character sets can be selected with the OPTION COUNTRY command.

The ASCII character code table is shown in Appendix F, together with a list of differences between the U.S. ASCII character set and those of the other eight countries.

† The byte is the unit in which data is handled by the PX-8's central processing unit (CPU); one byte consists of eight bits, or binary digits. In the binary numbering system, which uses the numerals 0 and 1, it is possible to represent all numbers from 0 to 255 as numbers of up to 8 digits. This is the range of numbers which is used for representing characters in the ASCII character code system.

2.6.2 Control characters

The ASCII character set includes a number of special codes which can be used in programs to control various devices. These control codes have different functions when used to control the LCD screen or a printer. Also they have further special functions when used with the screen editor. The following table outlines the functions of the control characters when acting on just the LCD screen. They are used in a PRINT statement together with the CHR$ function. For example:

PRINT CHR$ (12)

clears the screen.
There are also extended control sequences which consist of a group of character codes following the ESC code (ASCII 27 decimal, 1B hexadecimal). These are described in Appendix C.

Dec. code	Hex. code	Function
5	05	Deletes characters to the end of the line.
7	07	Sounds the speaker (at about 440 Hz).
8	08	Moves the cursor to the left.
9	09	Moves the cursor to the next tab postion.
10	0A	Moves the cursor down one line.
11	0B	Moves the cursor to the home position.
12	0C	Clears the virtual screen.
13	0D	Moves the cursor to the beginning of the line.
16	10	In mode 0/1/2, moves the screen window upward.
17	11	In mode 0/1/2, moves the screen window downward.
26	1A	Deletes all characters to the end of the screen.
27	1B	Escape code.
28	1C	Moves the cursor to the right.
29	1D	Moves the cursor to the left.
30	1E	Moves the cursor upward.
31	1F	Moves the cursor downward.

2.6.3 Numeric data

All numeric data is converted to binary form for storage in memory or calculation by the PX-8's CPU. However, BASIC allows numeric data to be entered in any of three number bases. These are decimal (base 10), octal (base 8), and hexadecimal (base 16).

Decimal notation is the familiar numbering system we use in everyday life, with numerals which range from 0 to 9. With this system, the number of digits required to express numbers increases by one each time the magnitude of the number being expressed increases by a factor of ten (1, 10, 100, and so forth.) Decimal notation can be used to represent both integers and numbers which include decimal fractions.

Octal notation (also referred to as base 8 numeration) uses only the digits from 0 to 7. With this system, the number of digits required to express numbers increases by one each time the magnitude of the number being expressed increases by a factor of eight. Octal numbers are indicated by affixing the characters "&O" or "&" to the beginning of the number. The decimal equivalents of octal numbers can be calculated as shown below.

$$\&O347 = 3\times8^2 + 4\times8^1 + 7\times8^0 = 231$$
$$\&1234 = 1\times8^3 + 2\times8^2 + 3\times8^1 + 4\times8^0 = 668$$

Numbers entered in octal notation may not include a decimal point; therefore, octal notation can only be used to represent integer values.

Hexadecimal notation (also referred to as base 16 numeration) uses the digits from 0 to 9 and the characters from A to F to represent the values from 10 to 15. With this system, the number of digits required to express numbers increases by one each time the magnitude of the number being expressed increases by a factor of 16. Hexadecimal numbers are indicated by affixing the characters "&H" to the beginning of the number. The decimal equivalents of hexadecimal numbers can be calculated as follows.

$$\&H76 = 7\times16^1 + 6\times16^0 = 118$$
$$\&H32F = 3\times16^2 + 2\times16^1 + 15\times16^0 = 815$$

As with octal notation, hexadecimal notation can only be used to represent integer values.

Ordinarily, decimal notation is used for display of the contents of memory or the results of calculations. However, it is also possible to specify display of integer values in hexadecimal or octal notation.

2.7 Constants — String Constants and Numeric Constants

Constants are fixed values which are writen into a program and used by that program during its execution. These values may consist of either characters or numbers; in the former case they are referred to as string constants, and in the latter case as numeric constants.

2.7.1 String constants

A string constant is any sequence of alphanumeric characters which is enclosed in quotation marks. Some examples of string constants are shown below.

"EPSON PX-8"
"John Jones"
"$10,000.00"
"The quick brown fox jumped over the lazy yellow dog."

The length of a string constant cannot exceed the maximum length of a program line (255 characters).

2.7.2 Numeric constants

Numeric constants are positive or negative numbers. There are five types of numeric constants, as follows:

(1) Integer constants
Integer constants are whole numbers in the range from −32768 to +32767. Such constants can be expressed in either decimal, hexadecimal, or octal form.

(2) Fixed point constants
Fixed point constants are positive or negative numbers which include a decimal fraction.

(3) Floating point constants
Floating point constants are positive or negative numbers which are represented in exponential form. A floating point constant consists of an integer or fixed point constant, followed by the letter E (denoting an implicit base of 10) and an exponent. Either the fixed-point part or the exponent may be preceded by a minus ("−") or plus ("+") sign to indicate that it is positive or negative; if no sign is present, it is assumed that that portion of the constant is positive. The exponent must be in the range from −38 to +38.

Example

$235.988E\text{-}Ø7 = 235.988 \times 10^{-7} = .0000235988$
$2.359E09 = 2.359 \times 10^{9} = 2359000000$

(With double precision floating point constants, the letter "D" is used to indicate the implicit base (10) instead of "E". See below for a discussion of single and double precision numeric constants).

2.7.3 Single and double precision numeric constants

PX-8 BASIC allows use of both single and double precision numbers. Single precision numbers are handled internally as seven significant digits, and are rounded to 6 digits for display or printout. Double precision numbers are handled internally as 16 significant digits, and are also printed or displayed as 16 digits (with leading zeroes suppressed).

A single precision constant is any numeric constant that fulfills one of the following conditions:

(1) Consists of seven or fewer digits;
(2) Is represented in exponential form with "E"; or
(3) Has a trailing exclamation point ("!").

A double precision constant is any numeric constant that fulfils one of the following conditions:

(1) Consists of eight or more digits;
(2) Is represented in exponential form with "D"; or
(3) Has a trailing number sign ("#").

NOTE:
The "#" character can be assigned different values, depending on the country in which the computer is being used. Consequently it will correspond to whatever character is assigned to CHR$(35).

Examples

Single Precision Constants	Double Precision Constants
46.8	345692811
−7.09E-06	−1.09432D-06
3489.0	3489.0#
22.5!	7654321.1234

NOTE:
When a BASIC program is written, care should be taken to declare the constants correctly. The BASIC interpreter may list numbers in a different form than that in which they are typed in. BASIC may also insert the trailing signs in the listing. For example the following lines:

```
10 PRINT A*1234567
20 PRINT B*123456789
```

would be listed as:

```
10 PRINT A*1.23457E+06
20 PRINT B*123456789#
```


2.8 Variables

Variables are named locations in memory which are used to hold values during execution of BASIC programs. Names are assigned to variables by the programmer, and the values stored in variables are either assigned by the user during program execution or assigned as a result of progam execution itself.

The two general types of variables used with BASIC are numeric variables and string variables. The former are used to store numeric values, and the latter are used to store character strings. Until a variable is defined it has the value of zero if it is a numeric variable, and the value of null or empty string if it is a string variable.

2.8.1 Variable names and type declaration characters

Variable names may consist of up to 40 characters (including all letters, the decimal point, and all numerals), followed by a type declaration character; however, the first character of each name must be a letter. Reserved words may not be used as variable names. (Reserved words are the keywords used in entering BASIC commands, statements, and functions). Further, the letters "FN" must not be used at the beginning of any variable name (BASIC interprets words beginning with the letters "FN" as calls to a user-defined function).

The names of string variables must end with a dollar sign ($); this is the type declaration character which indicates that a variable is used to hold string data.

Numeric variable names may end with type declaration characters which indicate the type of numeric data which they contain. The type declaration characters for numeric variables are as follows:

%	Integer variable type declaration character
!	Single precision variable type declaration character
#	Double precision variable type declaration character

A single precision numeric variable is assumed if no type declaration character is specified.

Examples of variable names are shown below.

PI#	Double precision numeric variable
MINIMUM!	Single precision numeric variable
LIMIT%	Integer variable
CATEGORY$	String variable

Variables may also be defined in advance as string, integer, single precision, or double precision with the DEFSTR, DEFINT, DEFSNG, and DEFDBL statements. When variable types are specified in this manner, type declaration characters are not required. See the explanations of these statements in Chapter 4 for details.

NOTE:
As with constants, BASIC may add the trailing declaration character when the program is listed. It is important to be aware that A# is a different variable from the variable A, unless A has been declared as a double precision variable with the DEFDBL statement. Also, statements declaring variable types do so for all variables beginning with a particular character and it is not possible to specify variable names consisting of more than one character in such statements.

2.8.2 Array variables

An array is a group of variables which is referred to by a common name. Each variable of an array is identified by one or more subscripts, each of which is specified as an integer value. The number of subscripts corresponds to the number of variables in a one-dimensional array (for instance, P(x) where "x" is the integer expression which identifies the individual variable); P(x, y) refers to a specific variable in a two-dimensional array, and can be thought of as a table containing a certain number of rows and columns; the number of rows depends on the maximum value of x, and the number of columns depends on the maximum value of y. Theoretically, an array can have any number of dimensions; however, in practice the number of dimensions and the size of the array are limited by the amount of memory which is available. The DIM statement is used to define the number of dimensions of an array and the size of each dimension. See the explanation of the DIM statement in Chapter 4 for details.

2.9 Type Conversion of Numeric Values

BASIC automatically converts numeric values from one type to another as necessary. This section describes the rules governing numeric type conversion for various kinds of operations.

(1) Type conversion upon storage of values in variables
If a numeric constant of one type is assigned to a numeric variable of another type, it is stored after being converted to the type declared for that variable name. For example, if an integer-type numeric constant is assigned to a single precision variable, it is automatically converted to a single precision value at the time it is stored. Note that a certain amount of error may be introduced by the process of conversion.

Example 1

```
10 A%=12.34        :'Assigns single precision number
20 '               :'12.34 to integer variable A%.
30 '
40 PRINT A%        :'DISPLAYS CONTENTS OF VARIABLE A%
Ok
run
 12
Ok
```

Example 2

```
10 A#=12.34        :'Assigns single precision number
15                 :'12.34 to double precision variable A#.
20 '
25 B#=23.45#       :'Assigns double precision number
30                 :'12.34# to double precision variable B#.
35 '
40 PRINT A#        :'Displays contents of variable A#.
45                 :'Extra digits are result of conversion
50                 :'error.
55 '
60 PRINT B#        :'Displays contents of variable B#.

Ok
run
 12.34000015258789
 23.45
Ok
```

(2) Conversion in arithmetic and relational operations

If an arithmetic or relational expression includes numeric opèrands of different types, all operands are converted to the same degree of precision (that of the operand with the highest degree of precision). Further, the results of arithmetic expressions are returned to the degree of precision of the most precise operand. Note that error may be introduced when constants are converted from one precision to another. (Note: Relational operations are described in section 2.10.2 below.)

Example

```
10 A#=6#/7.1#          :'Assigns result of arithmetic
20                     :'operation 6#/7.1# (double precision)
30                     :'to double precision variable A#.
40 '
50 PRINT A#            :'Displays contents of variable A#.

Ok
run
 .8450704225352113
Ok
```

In the example above, arithmetic is performed using double precision numbers and the result is returned in double precision.

Example

```
10 A#=6#/7.1    :'Assigns result of 6#/7.1 to variable A#.
20 '
30 PRINT A#     :'Displays contents of variable A#.
Ok
run
 .8450704338862249
Ok
```

Here, the single precision value 7.1 is converted to double precision for arithmetic and the result is returned as a double precision number. The difference between the result returned in this example and that returned in the preceding example is due to conversion error.

(3) Conversion for logical operations

During logical operations, non-integer operands are converted to integers and the result is returned as an integer. The operands of logical operations must be in the range from −32768 to 32767; otherwise an "Overflow" error will occur.

Example

```
10 PRINT 6.34 OR 15    :'Converts single precision
20                     :'number 6.34 to an integer
30                     :'value (6), then displays the
40                     :'logical sum of 6 and 15.
Ok
run
 15
Ok
```

See section 2.10.3 for an explanation of logical operations.

(4) Type conversion of floating point numbers to integers

When a floating point number is converted to an integer, the decimal fraction is rounded to the nearest whole number.

Example

```
10 C%=55.88     :'Converts single precision number 55.88
20              :'to integer by rounding to 56, then
30              :'stores 56 in integer variable C%.
40 '
50 PRINT C%     :'Displays contents of variable C%.

run
 56
Ok
```

(5) Conversion of single precision numbers to double precision

If a single precision number is assigned to a double precision variable, only the first seven digits of the converted number are significant. This is because only six digits of accuracy are provided by single precision numbers.

When evaluating a function care must be taken to ensure that the value of the expression being evaluated is declared to the precision required. In the following example, only the first seven characters of C# are correct because the square root of a single precision number is being converted to a double precision number. D# gives a more accurate value because the square root of a double precision number is being assigned to a double precision variable. Lines 80 and 90 show the result of printing a function without and then with double precision declarations.

Example

```
10 A=2
20 B#=2
30 C#=SQR(A)
40 D#=SQR(B#)
50 PRINT "C# =";C#
60 PRINT "D# =";D#
70 PRINT "The square root of 2 is";SQR(2#)
80 PRINT SQR(1+1)
90 PRINT SQR (1#+1#)

run
C# = 1.414213538169861
D# = 1.414213562373095
The square root of 2 is 1.414213562373095
 1.41421
 1.414213562373095
Ok
```


2.10 Expressions and Operations

An expression is any notation within a program which represents a value. Thus variables, numeric constants and string constants constitute expressions, either when they appear alone or when combined by operators with other constants or variables.

Operators are symbols which indicate mathematical or logical operations which are to be performed on given values. The types of operations which are performed by BASIC can be divided into four categories as follows:

(1) Arithmetic operations
(2) Relational operations
(3) Logical operations
(4) Functional operations

2.10.1 Arithmetic operations

The arithmetic operations performed by BASIC include exponentiation, negation, multiplication, division, addition and subtraction. The precedence of these operations (the order in which they are performed when included in a single arithmetic expression) is as shown below.

Operator expression	Operation	Sample
∧	Exponentiation	X∧Y
−	Negation (conversion of the sign of a value)	(−Y)
*, /	Multiplication, division	X*Y, X/Y
MOD	Modulus arithmetic	X MOD Y
\	Integer division	X\Y
+, −	Addition, subtraction	X+Y, X−Y

The concepts of integer division and modulus are explained in (1) and (2) below.

The order in which operations are performed can be changed by including parts of expressions in parentheses according to the normal rules of algebra. When this is done, the operations within parentheses are performed first according to the normal rules of precedence.

Sample algebraic expressions and their equivalents in BASIC are shown below.

Algebraic Expression	BASIC Expression
X+2Y	X+2*Y
X−Y÷Z	X−Y/Z
X×Y÷Z	X*Y/Z
$(X^Y)^2$	(X∧Y)∧2
X^{Y^2}	X∧(Y∧2)
X×(−Y)	X*(−Y)

When two consecutive operators are included in an expression, they should be separated by enclosure in parentheses as shown in the last example above.

(1) Integer division
With integer division, the operands of an expression are rounded to integers, then division is performed and the integer portion of the quotient is returned. The operator for integer division is the backslash (\). This should not be confused with standard division for which the operator is the slash (/).

NOTE:
Regardless of the International character set used, internal code CHR$(92) is used as the operator for integer division.

The following example of integer division compares the same division taking the integral part of the result after the division:

```
10 A=33.55:B=4.62
20 ID=A\B                :'Integer division
30 ND%=A/B               :'Normal division
40 PRINT ID,ND%
run
 6             7
Ok
```

When integer division is performed, both operands must be within the range −32768 to 32767.

(2) Modulus arithmetic
Modulus arithmetic is the arithmetic operation which returns the remainder of integer division as an integer. Rounding up can occur as the second example shows. The operator for modulus arithmetic is MOD. The precedence of modulus arithmetic is just below that of integer division.

Examples

```
PRINT 10 MOD 4
 2
Ok
PRINT 25.68 MOD 6.99
 5
Ok
```

(3) If division by zero is encountered during evaluation of an expression, the "Division by zero" message is displayed, machine infinity (the value of greatest magnitude which can be displayed by the computer) is displayed as the result, then execution continues.

Example

```
10 B=7/0                          :'Generates "Division by zero"
20                                :'error, then stores machine
30                                :'infinity in variable B.
40 '
50 PRINT"PROGRAM LINE 30" :'Displays "PROGRAM LINE 30".

run
Division by zero
PROGRAM LINE 30
Ok
```

The "Division by zero" message is also displayed and machine infinity returned when zero is raised to a negative power.

(4) An overflow error is the condition which occurs when the magnitude of the result of an operation exceeds the maximum value which can be displayed by the machine or when one of the operands of an operation such as integer division exceeds the maximum allowed value. Whether or not execution continues depends on which situation is encountered.

Example

```
10 A#=666^666                     :'Generates "Overflow"
20 '                              :'error and stores machine
30 '                              :'infinity in variable A#.
40 '
50 PRINT A#                       :'Displays contents of A#.
60 '
70 PRINT "Program line 30" :'Displays "Program line 30".
80 '
```

```
90 A%=66666!\25          :'Generates an "Overflow" error
100                      :'because dividend (66666) is
110                      :'outside of permitted range
120                      :'for integer division.
125
130 PRINT A%             :'Not executed because
140                      :'program execution is
150                      :'aborted by error in
160                      :'line 90.

run
Overflow
 1.701411733192645D+38
Program line 30
Overflow in 90
Ok
```

2.10.2 Relational operations

Operations in which two values are compared are referred to as relational operations. The result returned by such a comparison is either "true" (−1) or "false" (0), and is then used to make a decision regarding subsequent program flow. (See the discussion of the IF...THEN...ELSE and IF...GOTO statements in Chapter 4.)

The relational operators and their meanings are listed below.

Operator	Relation tested	Example expression
=	Equality	X=Y
< >, ><	Inequality	X< >Y, X>< Y
<	Less than	X<Y
>	Greater than	X>Y
<=, =<	Less than or equal to	X<=Y, X=<Y
>=, =>	Greater than or equal to	X>=Y, X=>Y

NOTE:
The "=" sign is used both for testing equality in relational expressions and in LET statements for assigning values to variables. However, its meaning is not the same in both cases. See the discussion of the LET statement in Chapter 4 for details on assigning values to variables.

When arithmetic and relational operators are combined in one expression, the arithmetic is always performed first. For example, the following expression is true if the value of X plus Y is less than the value of T−1 divided by Z.

$$X+Y<(T-1)/Z$$

Example

```
10 A=1=1
20 PRINT A
30 B=3>4
40 PRINT B
50 PRINT 3>2

run
-1
 0
-1
Ok
```

In the example above, line 10 tests for equality between the first and second operands of the relational expression "1=1", then stores the result (−1, or true) in variable A. Line 20 then displays the contents of A. Line 30 tests whether the first operand of the relational expression "3>4" is greater than the second, then stores the result (0, or false) in variable B. The result is then displayed by the statement on line 40. Line 50 evaluates and displays the result of the relational expression "3>2" (−1, or true).

2.10.3 Logical operations

A logical operation uses Boolean arithmetic to define the logical connection between the results (−1 or 0) of relational operations. In any given expression, logical operations are always performed after arithmetic and relational operations. The results of operators are listed in the table below according to the order of precedence.

NOT (Negation)

X	NOT X
1	0
0	1

AND (Logical product)

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

OR (Logical sum)

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

XOR (Exclusive − OR)

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

IMP (Inclusion)

X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1

EQV (Equivalence)

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

Since relational operations can be used to make decisions concerning program flow, logical operators can be used to connect two or more relational operations. This allows decisions to be based on multiple conditions. (See the discussion of the IF...THEN...ELSE and IF...GOTO statements in Chapter 4).

Examples

1) IF D<200 AND F<4 THEN 80
   This statement causes program execution to branch to line 80 if the contents of variable D are less than 200 and the contents of variable F are less than 4.
2) IF I<10 OR K<0 THEN 50
   This statement causes program execution to branch to line 50 if the contents of variable I are less than 10 or the contents of variable K are less than 0.

In a logical operation, the operands are converted to signed 16-bit two's complement integers† in the range from −32768 to 32767 before their logical connection is checked. (An error will result if any operand is not within this range.)

The specified operation is then performed for each bit of each operand (that is, for bits which are in the same position in each operand) and the result is returned as a two's complement integer which represents the results for all bits. Some examples of this are shown below.

Example 1

```
10 PRINT 63 AND 16
Ok
run
 16
Ok
```

In binary notation, the two's complement integer 63 is 111111B and the two's complement integer 16 is 010000B. Since 1 AND 0 yields 0 and 1 AND 1 yields 1, the result is 010000B, or 16.

Example 2

```
10 PRINT 21 XOR 17
Ok
run
 4
Ok
```

The two's complement integer 21 is expressed in binary as 10101, while the two's complement integer 17 is expressed as 10001; since 1 XOR 1 and 0 XOR 0 yield 0, while 1 XOR 0 yields 1, the result is 00100, or 4.

Example 3

```
10 PRINT -1 OR -2
Ok
run
-1
Ok
```

† The first bit of a two's complement integer indicates whether the integer is positive or negative. In binary notation, the two's complement integers from 0 to 32767 are expressed as 0000000000000000B to 0111111111111111B. The integers from −1 to −32788 are expressed as 1111111111111111B (−1) to 1000000000000000B (−32768). The value 1111111111111111B is obtained by adding 1 to the complement of 0000000000000001B (i.e., 1111111111111110B+1B=1111111111111111B). The binary representations of other negative two's complement integers can be obtained in the same manner.

The two's complement integer −1 is expressed in binary as 1111111111111111B, while the two's complement integer −2 is expressed as 1111111111111110B. Since both 1 OR 1 and 1 OR 0 yield 1, the result is 1111111111111111B, or − 1.

Logical operators can be used to test data bytes for a particular bit pattern. For instance, the AND operator can be used to mask all but one bit of a status byte to obtain the status of a device I/O port; or, the OR operator can be used to merge two data bytes to obtain a particular value.

Example

```
10 FOR I=97 TO 122          :'Displays characters
20 PRINT CHR$(I);           :'from "a" to "z".
30 NEXT                     :'
40 '
50 PRINT                    :'Moves cursor down to
60                          :'next line on display.
70 '
80 '      Following lines convert lowercase
90 '      character codes to uppercase.
100 FOR I=97 TO 122         :'Binary  nnnnnnnn
110 PRINT CHR$(I AND 223);:'AND     11011111
120 NEXT                    :'yields: nn0nnnnn

run
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ok
```


2.10.4 String operations

String operations involve manipulation of character strings with operators. For example, the "+" operator makes it possible to concatenate (link) strings as shown in the example below.

Example

```
10 A$="File":B$="name"  :'Assigns string "File" to A$
20                      :'and string "name" to B$.
30 '
40 PRINT A$+B$          :'Concatenates string
50                      :'expressions A$ and B$ and
60                      :'displays the result.
70 '
80 PRINT "New "+A$+B$   :'Concatenates string
90                      :'expressions "New ", A$,
100                     :'and B$ and displays the
110                     :'result.

run
Filename
New Filename
Ok
```

Character strings can also be compared using the same relational operators as are used with numeric values.

```
10 A$="ALPHA":B$="BETA"
20        :'Assigns string "ALPHA" to A$ and
30        :'string "BETA" to b$.
40 '
50 IF A$<B$ THEN 100 ELSE 120
60        :'Jumps to line 100 if value of A$ is less
70        :'than that of B$; otherwise, jumps to
80        :'line 120.
90 '
100 PRINT A$;" IS LOWER THAN ";B$
110 END   :'Stops program execution.
120 PRINT A$;" IS NOT LOWER THAN ";B$

run
ALPHA IS LOWER THAN BETA
Ok
```

Strings are compared by taking one character at a time from each string and comparing their ASCII codes. The strings are equal if all codes are the same; if the codes differ, the character with the lower ASCII code is regarded as lower.

Comparison ends when different characters are encountered in the two strings or when the end of one of the strings is reached; in the former case, the string in which the lesser code is encountered is regarded as smaller, and in the latter case the shorter string is regarded as smaller.

Spaces included in strings are also significant; for example:

"ALPHA" is smaller than "ALPHA "
"ALPHA" is greater than " BETA"

Further examples are:

"AA" is less than "AB"
"FILENAME" is equal to "FILENAME"
"X&" is greater than "CL"
"SMYTH" is less than "SMYTHE"

Thus, string comparisons can be made to test string values and to sort strings into alphabetical order. All string constants used in relational expressions must be enclosed in quotation marks.

2.11 Functions

Functions are operations which return a specific value for a single operand. For example, the function SIN(X) returns the sine of the numeric value stored in variable X when the value in X is in radians. A variety of functions are built into PX-8 BASIC; these are referred to as intrinsic functions, and are described in Chapter 4.

PX-8 BASIC also allows the programmer to define his own functions; for details, see the explanation of the DEF FN statement in Chapter 4.

2.11.1 Integer functions

The CINT, FIX, and INT functions all return an integer value for an argument consisting of a numeric expression. These functions are described in detail in Chapter 4.

(1) CINT
The CINT function rounds the argument to the nearest integer value.

Examples: CINT(1.1) = 1
CINT(0.9) = 1
CINT(−5.4) = −5
CINT(−5.7) = −6

(2) FIX
The FIX function truncates the argument; that is, it discards the decimal portion.

Examples: FIX(1.1) = 1
FIX(0.9) = 0
FIX(−5.4) = −5
FIX(−5.7) = −5

(3) INT
The INT function returns the largest integer which is less than or equal to the argument.

Examples: INT(1.1) = 1
INT(0.9) = 0
INT(−5.4) = −6
INT(−5.7) = −6

2.11.2 Trigonometric functions

PX-8 BASIC supports the following trigonometric functions.

Function	Argument	Value returned
ATN	Tangent of an angle	Angle in radians
COS	Angle in radians	Cosine of an angle
SIN	Angle in radians	Sine of an angle
TAN	Angle in radians	Tangent of an angle

If you want to work with angular measurements in degrees, remember that you will have to convert the arguments of these functions from degrees into radians (with the ATN function, you will have to convert the result from radians into degrees). Since 180 degrees is equal to π radians, there are $180/\pi$ degrees in a radian. Thus, you can convert degrees to radians by dividing by $180/\pi$. Conversely, radians can be converted to degrees by multiplying by $180/\pi$. Single and double precision values corresponding to $180/\pi$ are as follows.

Single precision
180/3.1416=57.2958

Double precision
180/3.141592653589795=57.29577951308228

Other trigonometric functions must be derived from these four built-in functions. For example, the secant of an angle is equal to 1 divided by the angle's cosine. See Appendix E for derivation of other trigonometric functions.

2.12 Files

In general, a file is any set of data records which is output to or input from an external device (such as a disk drive) under a common identifier. This includes text files containing the program lines of BASIC programs, machine language program files and data files. Files can be stored in the RAM disk, microcassette tape or on floppy disks; however, they may also be input from and output to other devices. (See Chapter 5 for information on the types of file organizations used with BASIC for the PX-8.)

2.12.1 File descriptors

With PX-8 BASIC, files are identified by means of descriptors which consist of a device name and a file name. Together, these are referred to as the "file descriptor" and are specified as follows.

<device name> <option> <file name>

(1) Device name
PX-8 BASIC supports the concept of general device I/O. This means that input and output access to all devices can be handled in the same manner, regardless of whether the device accessed is a floppy disk drive, printer, the microcassette drive, or the RS-232C interface.

The format of all input and output commands is the same regardless of the type of I/O device. I/O devices are distinguished from one another by means of device names; the devices which can be addressed for I/O operations in this manner are as follows.

Name	Device	Modes
KYBD:	Keyboard	Input only
SCRN:	LCD screen	Output only
LPTØ:	Printer	Output only
COMØ:	RS-232C interface	Input and output
A: to H:	Disk devices	Input and output (Input only for ROM capsule)

Device names may be omitted when specifying file descriptors; however, the current CP/M default device is assumed if the device name is omitted.

NOTE:
Device name LPT0: can be assigned to either the RS-232C port or the serial port with the CONFIG program of CP/M to indicate the printer connected to that port. Initially, LPT0: is assigned to the RS-232C port.

(2) Option
<option> is used to set the baud rate and communication format for the RS-232C interface, the write mode for the microcassette drive, and so forth. The format of <option> varies according to device.

(3) File name
File names are used to distinguish files within a device from one another. Specification of file names is mandatory when accessing files in disk devices; however, file names are meaningless in the case of device files such as the keyboard and RS-232C interface, and will be ignored if specified.
A <file name> is composed of a primary name of up to 8 alphanumeric characters, and an extension consisting of up to 3 alphanumeric characters. The primary file name is separated from the extension by a full stop.

Example SAMP1.BAS

With the LOAD, MERGE, RUN, LIST and SAVE commands, the extension ".BAS" is assumed only if the primary name is specified in the command's operand. The FILES, KILL or NAME commands require extensions to be specified.

2.12.2 File numbers

A file number is a number which is assigned to a device file as an identifier when that device is opened for input and/or output. The number specified as the file number must be in the range from 1 to the maximum specified in the /F: option when BASIC operation is started.
With PX-8 BASIC, a logical file number must be assigned to each file which is read or written by a program (except when a program text file is accessed using the LOAD, MERGE, RUN, LIST or SAVE commands). This is done with the OPEN statement, which links a specific logical file number to the physical file defined in the file descriptor. Unless otherwise specified with the /F: option when BASIC is started, the maximum number of files which can be open at one time is 3. See the explanation of the OPEN statement in Chapter 4 for the procedure for assigning file numbers.

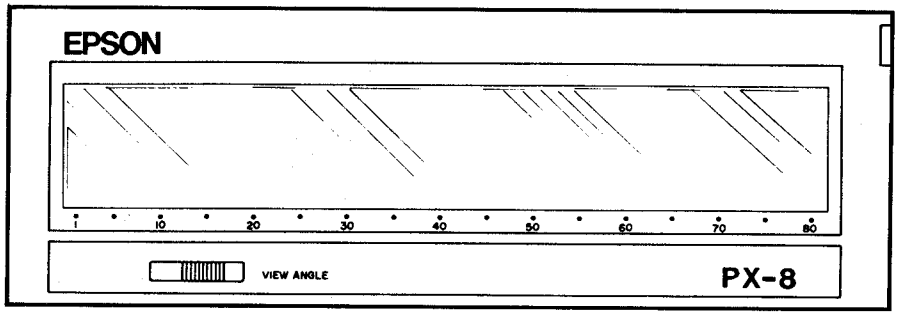
2.13 Display Screen

This section discusses the types of screens used with PX-8 BASIC, describes the various screen modes and the manner in which they are used, and explains the systems of coordinates which are used in specifying the locations of characters and graphics on the screen.

2.13.1 Real screen, virtual screen, and virtual screen window

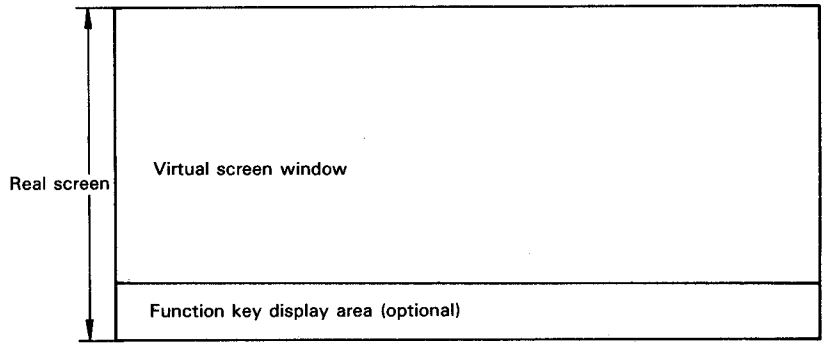
With PX-8 BASIC, several different screen concepts are used for display. These include the real screen, virtual screens and the virtual screen window .

(1) Real screen
The PX-8 LCD screen permits display of up to 8 lines of 80 characters each or 480 by 64 dots of graphics. The LCD device on which characters and graphics are physically displayed is referred to as the real screen.



(2) Virtual screen
Although the LCD screen of the PX-8 allows display of up to 8 lines of 80 characters each, the concept of virtual screens has been introduced to make it possible to use application programs which require screens with even larger capacities. A virtual screen is not a physical device like the real screen, but exists in an area in memory which is referred to as VRAM (video random access memory) and is displayed through the "window" provided by the real screen. This VRAM is connected to the PX-8's 6301 slave CPU, and cannot be accessed by the PEEK or POKE commands of BASIC. Except in the graphics screen mode there are two virtual screens. The two virtual screens are independent of one another and either can be displayed under program control. In the split screen modes, it is possible to display both virtual screens on the two halves of the real screen.

(3) Virtual screen window
Since the real screen is limited to seven or eight lines depending on whether the function key definitions are displayed, it acts as a window on the virtual screens. However, scrolling can only be performed in the vertical direction, either by means of the cursor keys or under program control.



When function key definitions are not displayed, the number of lines displayed by the virtual screen window (also referred to as the screen window) is the same as the number of lines in the real screen (8). When function key definitions are displayed, the number of lines displayed is reduced by one (the line used to display function key definitions).

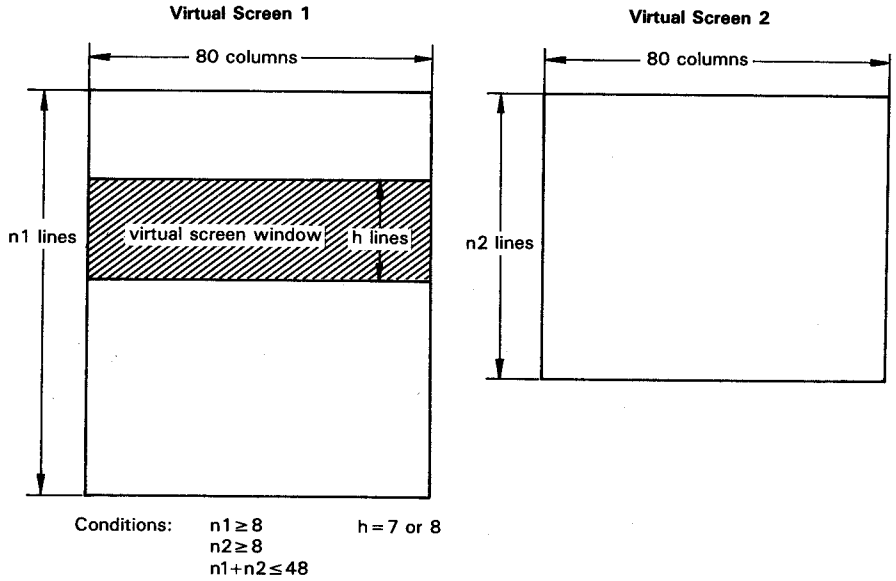
2.13.2 Screen modes

The PX-8's LCD screen has four modes of operation. These are referred to as screen modes 0, 1, 2, and 3. Screen modes 0, 1, and 2 are solely text screen modes; screen mode 3 is the graphic display mode; and allows text to be mixed with graphics.

When the display is in one of the text modes, the PX-8's VRAM is divided into two sections which are used as two independent virtual screens. In the text modes, character data consisting of one-byte ASCII codes is written into VRAM; these codes are converted to character images for display by the PX-8's display controller.

When display is in the graphic screen mode, all of VRAM is used for storing the settings (on or off) of all the dots in the LCD screen, rather than codes representing character data.

(1) Screen mode 0 (the 80-column text screen mode)
In this screen mode, the two virtual screens each have a width of 80 columns (characters per line). The number of lines in each of the virtual screens can be set as desired by the user, provided that the total number of lines in the two screens does not exceed 48 and that each screen contains at least as many lines as the real screen. The virtual screen window can be switched back and forth between the two virtual screens, and can be scrolled up or down in the currently selected virtual screen to display its contents.

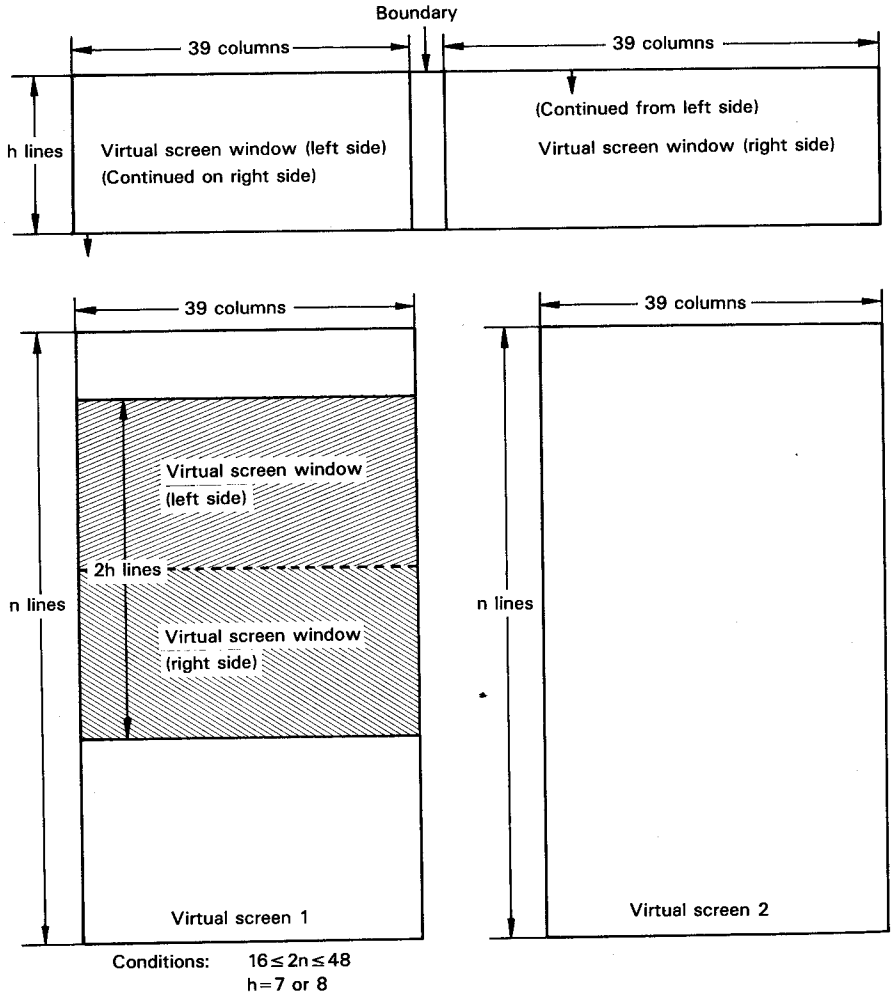


(2) Screen mode 1 (the 39-column text screen mode)
In this screen mode, the virtual screen window is split vertically into two parts, each with a width of 39 columns. The remaining two columns of the 80 making up the real screen are used to display characters which represent a boundary between the two halves.

Both sides of the virtual screen window are positioned over the same virtual screen, and the display at the bottom of the left half of the virtual screen window is continued at the top of the right half.

In this screen mode, the size of the virtual screen window is 39 characters × 2h lines (where h is the number of lines in each half of the screen window).

As with screen mode 0, the virtual screen window can be switched back and forth between the two virtual screens, both of which have a width of 39 columns. The number of lines in the two virtual screens can be set as desired by the user within the range from 16 to 48, and both virtual screens must have the same number of lines.

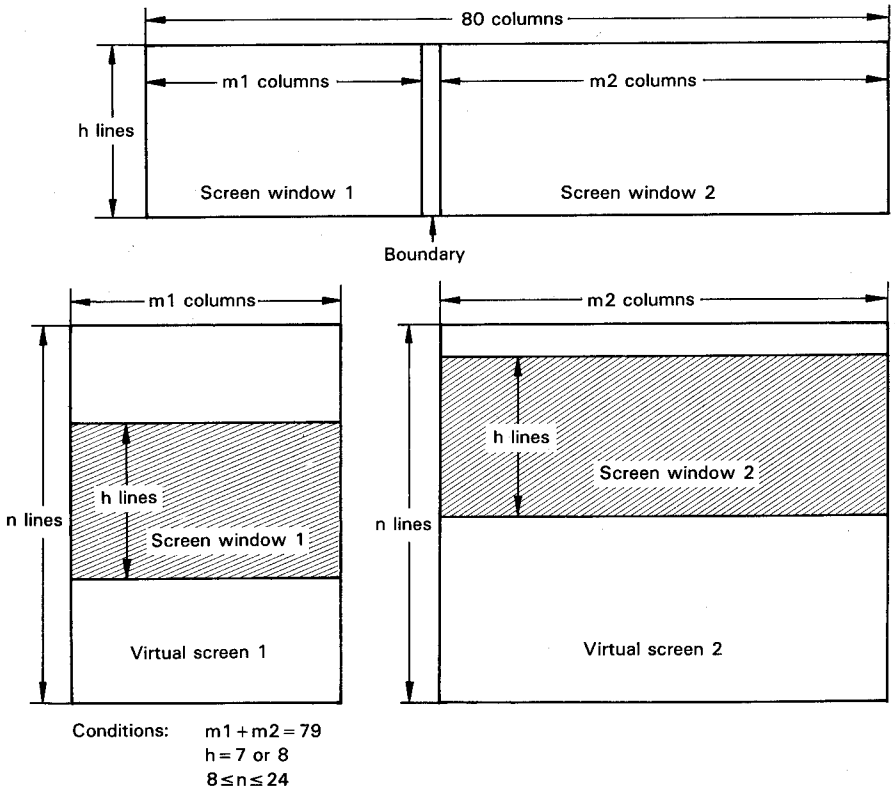


Conditions: $16 \leq 2n \leq 48$
$h = 7$ or 8

(3) Screen mode 2 (the dual screen mode)
In this screen mode, the virtual screen window is vertically divided into two parts, each of which displays the contents of one of the virtual screens and which can be scrolled independently of the other. This makes it possible to display the contents of both virtual screens at the same time.

The width of each part of the screen window can be set as desired by the user as long as the total number of columns in both parts is equal to 79. The remaining column of the real screen is used to display a boundary character between the two screen windows.

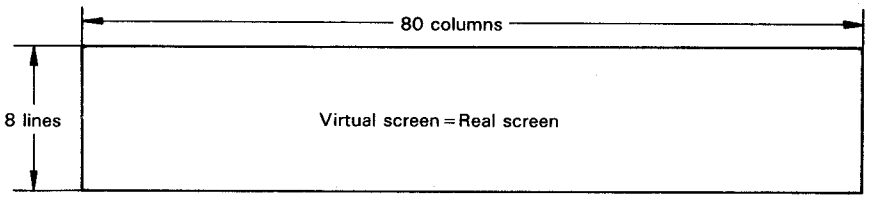
In this screen mode, the two virtual screens each have a column width which is equal to the width of the screen window which displays its contents. The number of lines in each virtual screen can be set as desired by the user in the range from 8 to 48; however, both virtual screens must have the same number of lines.



Conditions: $m1 + m2 = 79$
$h = 7$ or 8
$8 \leq n \leq 24$

(4) Screen mode 3 (the graphic screen mode)
This is the screen mode which is used for displaying graphics. In this screen mode, the PX-8's display controller works on a bit image basis, rather than using character codes, making it possible to display a full range of graphics. Although text can be displayed with the graphics, the size of the virtual screen in this screen mode is the same as the size of the real screen, that is, the virtual screen size is 80 columns × 8 lines. When the function key assignments are displayed, there are only 7 usable lines.

Although the screen editor can be used in the same manner in this screen mode as in the text screen modes, the virtual screen (i.e., bit image data in VRAM) is displayed directly to the real screen and there is no screen window. Because the real and virtual screens are the same, scrolling of the virtual screen cannot occur. Graphic statements such as PSET, PRESET, LINE and POINT can only be used in this screen mode.



2.13.3 Selection and display of virtual screens

Display screen modes 0 to 2 each include two virtual screens, only one of which can be selected at any given time. The screen selected is that in which characters are displayed when keys are pressed or when PRINT and similar statements are executed. The SCREEN command in BASIC is used to select the virtual screen.

(1) Screen modes 0 and 1
In these screen modes, only one virtual screen can be displayed at a time. The virtual screen selected is displayed in the screen window, and characters typed or output are displayed in the selected virtual screen. The scrolling control keys and scrolling escape sequences move the virtual screen window through the virtual screen which is currently selected.

(2) Screen mode 2
In this screen mode, both virtual screens are displayed at the same time. However, the scrolling control keys and scrolling escape sequences only move the virtual screen window through the virtual screen which is currently selected for output (the screen in which typed characters are displayed).

2.13.4 Scrolling control

The screen window has two scrolling modes, either of which can be selected by pressing the SCRN key (SHIFT + INS) or by escape sequence. These two modes are referred to as the tracking mode and the non-tracking mode.

(1) Tracking mode
In this mode, the screen window is scrolled along with the cursor (it "tracks" the cursor). If the cursor is outside the screen window when this mode is selected from the non-tracking mode, the screen window immediately moves to the position of the cursor in the virtual screen.

(2) Non-tracking mode
In this mode, the screen window does not follow the cursor.

The following keys are used for scrolling.

(1) SHIFT + ↑ (scroll up)
Pressing these keys together scrolls the screen window upward.

(2) SHIFT + ↓ (scroll down)
Pressing these keys together scrolls the screen window downward.

(3) CTRL + ↑ (top-of-screen)
Pressing these keys together moves the screen window to the top of the selected virtual screen.

(4) CTRL + ↓ (end of screen)
Pressing these keys together moves the screen window to the bottom of the selected virtual screen.

When the screen window is scrolled using the four keys described above, the cursor does not move from its current position in the virtual screen. This makes it possible to keep the cursor in one position while the screen is moved, even if scrolling is done in the tracking mode.

(5) SHIFT + INS (change scroll mode)
When the same virtual screen is used both as the write screen and the display screen, this key switches scrolling back and forth between in the tracking mode and the non-tracking mode. The scrolling mode used is switched each time this key is pressed.

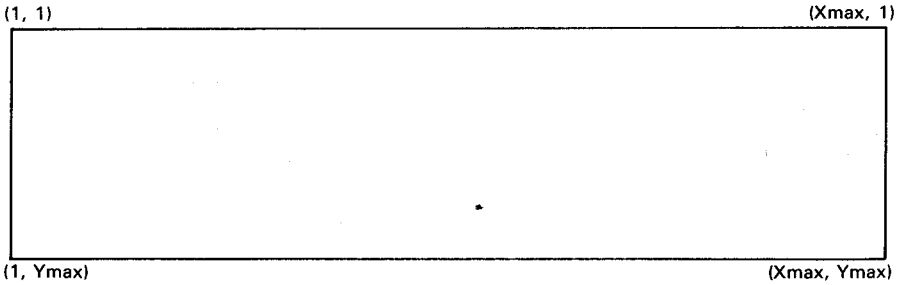
(6) CTRL + INS (find cursor)
When the cursor is not displayed in the screen window, pressing CTRL + INS moves the screen window to the current position of the cursor.

2.13.5 Screen coordinates

(1) Character coordinates
Character coordinates are the coordinates which are used to specify the position in which characters are displayed on the screen. These coordinates are used with the LOCATE statement and the SCREEN, POS and CSRLIN functions.

When using character coordinates, the column on the left side of the screen is numbered 1 and that on the right side of the screen is numbered according to screen mode or the maximum screen width specified by the user. In screen modes 0 and 3, the column on the right side of the screen is numbered 80 and in screen mode 1 it is numbered 39. In screen mode 2, the number of the right hand column is the same as the column width specified for the selected virtual screen by the user.

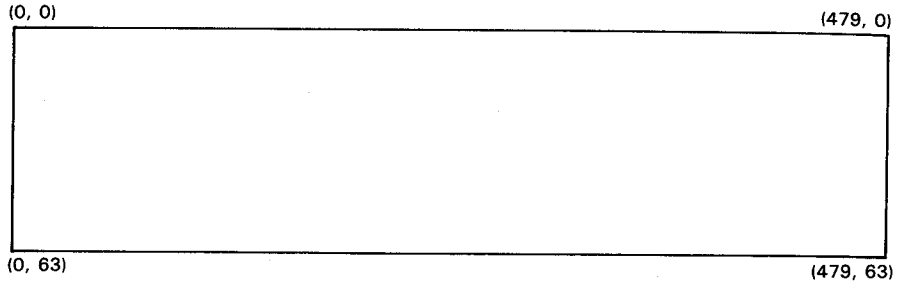
In the vertical direction, the top line is numbered 1 and the bottom line is numbered according to screen mode or the maximum number of screen lines specified by the user.



Xmax: Number of columns in selected virtual screen
Ymax: Number of lines in selected virtual screen

(2) Graphic coordinates
Graphic coordinates are used to specify the positions of individual dots on the screen. The graphic coordinate system is used with statements such as PSET, PRESET and LINE, and with graphic functions such as POINT.

With graphic coordinates, dots are numbered horizontally from 0 on the left side to 479 on the right, and vertically from 0 at the top of the screen to 63 at the bottom.



When the positions of individual dots on the screen are specified directly, the absolute form (<horizontal position>,<vertical position>) is used, for example PSET (19, 29) would switch on the twentieth dot across on the thirtieth row. This type of coordinate specification can be used with all graphic statements and functions.

It is also possible to specify the positions of screen dots in relation to previously specified dots; in this case, the coordinate specification takes the form STEP (<horizontal position>,<vertical position>). Here, STEP indicates that the values specified for (<horizontal position>,<vertical position>) are to be added to the values contained in a pointer called the last reference pointer or LRP which indicates the absolute coordinates of a previously specified dot. Relative coordinate specification can be used with the PSET, PRESET and LINE statements, and the last reference pointer (LRP) is updated by execution of these statements. Thus the following example will plot a row of ten dots at intervals of ten horizontal positions. Line 20 sets the position of the first point absolutely and so when line 40 is executed for the first time the LRP is the position (20, 10).

```
10 SCREEN 3
20 CLS
30 PSET (20,10)
40 FOR J=1 TO 9
50 PSET STEP (10,0)
60 NEXT J

Ok
```

2.14 A Practical Guide to the Screen Modes

Whereas the above description summarises the possibilities for the various screen modes, it is difficult to appreciate the full facilities without seeing them in action. This section is thus meant to be followed actually using the PX-8.

The various screen modes are accessed by using the SCREEN command. The screen size is set by the WIDTH command. This can be attached to the SCREEN command to give the full format of the screen command as follows:

SCREEN M,VS,FKS,BC WIDTH C,NL1,NL2

Where M is screen mode 0, 1, 2 or 3

VS is the virtual screen to be displayed.

FKS allows the function key assignments at the base of the screen to be switched on and off to give the full 8 lines of the LCD screen.

BC sets the boundary character for split screen display in screen mode 2.

WIDTH is a separate command which is used to set the number of columns and lines of the virtual screens. It would normally be used on its own, but since in many cases use of the SCREEN command will involve setting the size of the virtual screens, it has been added to the syntax of the SCREEN command. The addition does not require a comma to separate the WIDTH command from the boundary character, but it does require a space as separator. In WIDTH, C sets the number of columns (in screen modes 1 and 2) and NL1 and NL2 the number of lines of the two virtual screens. The range of these options varies according to the modes and thus will be discussed under each mode.

If one of the parameters is to be changed, other than the mode, the correct number of commas must be inserted to denote which parameter is being changed. To illustrate this the function key assignments can be switched off using the command

SCREEN,,0

because the function key switch is the third parameter.

If a screen mode or virtual screen is not specified, the current values are used.

The display on the LCD screen is a window on one or both of the virtual text screens. In mode 0, 1, or 3 only one of the virtual screens is displayed at a particular time. In mode 2 they can be displayed side by side.

If the screen mode is changed, the real and virtual screens will be cleared. However, if only the virtual screen is being changed, the window is changed and there is no clearing of either virtual screen. In illustrating this and successive operations, it is necessary to see the effect in each screen mode.

(1) MODE 0

The following sequence of operations shows how switching the display between the virtual screens leaves the virtual screens intact, and then goes on to illustrate how to move about the screens.

Type MENU and press [RETURN], or use the shifted function key [PF4].

From the BASIC menu login to a program area.

You will now be in screen mode 0 and see a portion of the first virtual screen. Without clearing the screen, type SCREEN 0,1 and press the [RETURN] key; the screen will clear but for "Ok" and a flashing cursor (this is the second virtual screen).

Now type SCREEN 0,0 and press [RETURN]; the first virtual screen will be redisplayed as you left it, except that the cursor will have moved down and another "Ok" will have been written onto the screen.

It is possible to switch screens in the direct mode using the [CTRL] and [←] or [→] cursor keys. If you hold down the [CTRL] key and press the [←] cursor key the display will switch to the first virtual screen. If the first virtual screen is being displayed, then no change will be apparent. Similarly, pressing the [CTRL] and [→] cursor key will show the second virtual screen. This can only be used in the direct mode and not in programs, even in INPUT statements.

The [↑] and [↓] cursor keys also have a function when pressed together with the [CTRL] key. They cause the first and last displayable lines respectively of the current virtual screen to be displayed. Depending on whether the screen is showing the function key assignments, 7 or 8 lines will be shown.

If the cursor is not on these lines it will not be displayed. Set the screen to the last lines of the virtual screen and type another key. If the key is not [CTRL],

SHIFT, CAPS LOCK or NUM/GRAPH or a key which cannot function (e.g., the BS key if the cursor is at the beginning of a line) the window will be returned to the part where the cursor is located. The character pressed will be printed beside the cursor, or the function of the key carried out. You can also return to the cursor position by pressing the CTRL and INS keys together.

The cursor can be moved anywhere on the virtual screen using the cursor keys. Note how the screen moves with the cursor if you move to a line above or below the real screen. This is the normal tracking mode. It can be changed to the non-tracking mode, where the cursor can be moved anywhere on the virtual screen, leaving the window fixed. Pressing the SCRN key (shifted INS) will switch between the two modes. Move the cursor off the real screen with the cursor keys in the non-tracking mode, and then restore the part of the screen containing the cursor to the window as before using the CTRL and INS keys. It is possible to set the cursor to the first character position of the virtual screen with the HOME key (shifted BS) but this does not display the cursor. In combination with the CTRL and INS keys as a sequence it can be used to set the display to the top of the virtual screen with the cursor on that position.

It is still possible to scroll the screen even if the cursor is not in the window, by using the SHIFT and ↑ and ↓ cursor keys. This only moves the screen up or down one line at a time.

The boundary character is not used in screen mode 0.

The WIDTH command is used to set the number of columns and number of lines of each virtual screen. However, in screen mode 0 it is not possible to alter the number of columns. It is only possible to display an 80 column screen, with either of the two virtual screens being displayed at any one time. If using screen mode 0 a value of 80 must be used for the column width or no value used at all.

The number of lines in each virtual screen can be altered provided the sum of the lines specified is less than or equal to 48, and that neither screen is made less than 8. If your program uses only one virtual screen, it is beneficial to increase its size to the maximum, (i.e., 40 since the other screen must be 8 and the total 48). It is also useful to do this if you are writing a BASIC program since it is possible to scroll back without having to relist the program. Type

SCREEN 0,0,0, WIDTH 80,40,8

or the equally valid

SCREEN 0,0,0, WIDTH ,40,8

This sets the first virtual screen to 40 lines and the second to 8. It also switches to the first virtual screen if you were not already displaying it, and turns off the function key assignments display on the bottom line. Note that there is no reference to boundary character, because there is no division of the screen in this screen mode, but the comma as separator must be inserted or an error will occur. You can see the extent of the first virtual screen by using the cursor keys. Now press the CTRL and → keys to display the second virtual screen. Now if you fill the screen with text, e.g. a listing, you can see that the cursor keys only allow movement on the real screen (because the virtual screen is the same size as the real screen).

(2) MODE 1

In changing to screen mode 1, use the SCREEN command to show the first virtual screen and turn the function key assignment display off by typing the following SCREEN command and pressing RETURN

SCREEN 1,0,0

The screen will clear to give a display with a boundary of two characters width in the centre of the creen. This boundary marker cannot be changed either in position or as the character displayed.

Use the cursor keys to move down the screen. You will see that when the cursor reaches the base of the left-hand side of the screen, instead of disappearing off the bottom and causing the screen to scroll up, it moves to the top right-hand part of the screen. Only when the cursor reaches the base of the right-hand side of the screen does the top line scroll off the left-hand side of the screen. Moreover, the right-hand side of the screen scrolls up into the left-hand side. This is because the two halves of the screen are displaying 16 lines of the virtual screen 39 columns wide but in two blocks of 8 lines side by side. This can be seen better if the following program is run to fill the screen with numbers on each line.

```
10 CLS
20 FOR J = 1 TO 20
30 PRINT J
40 NEXT
```

Try the following to see how it differs from screen mode 0.

(i) Change the window between the virtual screens, using CTRL and the → and ← cursor keys.
(ii) Change the tracking mode, using the SCRN key.
(iii) Scroll using the shifted ↑ and ↓ keys.
(iv) Look at the first and last displayable lines using the CTRL and ↑ and ↓ keys.

Only one parameter can be altered in the WIDTH statement in screen mode 1. The screen is set to display two 39-column halves on each side of the screen. It is not possible to alter this so the WIDTH statement must specify 39 as the column width, if any value is used, or an error will occur. The number of lines in each virtual screen is the same in screen mode 1. The number of lines must be specified in the range 16 to 48 and setting the number of lines for the first screen also sets the number for the second. Any attempt to set the size of the second virtual screen will be ignored and no error generated. Thus to set the number of lines on each screen to 20, the following are valid commands:

WIDTH 39, 2Ø
WIDTH ,2Ø
WIDTH 39, 2Ø, 8

(3) MODE 2

This is the most versatile screen mode. The width of each half of the screen and boundary character can be set. The number of lines on each virtual screen is the same as with screen mode 1. For example type

SCREEN 2, Ø, Ø, "*" WIDTH 2Ø, 4Ø

This will switch to screen mode 2, placing the cursor on the left-hand side of the screen and removing the function key assignments from the base of the screen. It also sets the width of the left-hand side of the screen to 20 columns. The right-hand side is thus set as 59 since one of the 80 columns is used for the boundary characters.

Now use the CTRL and → keys to change the virtual screen. Note how the cursor moves to the right half of the screen. This is because the two halves of the screen correspond to the two virtual screens. Use the cursor keys in combination with the SHIFT and CTRL keys to explore this mode as with modes 1 and 2.

The boundary character can be reset using an ASCII character code with the CHR$( ) function. For example type the following to change the character to the vertical line graphics character (ASCII code 134):

SCREEN ,,,CHR$(134)

Note how both of the virtual screens are cleared when this command is executed.

(4) MODE 3

This screen mode is the mixed graphics and text mode. The dots are individually addressable, using the various graphics commands. This means that the video memory is largely devoted to storing the graphics information. Consequently text is limited to one virtual screen. This virtual screen is the same size as the real screen (80 columns and 8 lines) and so the various combinations of CTRL and cursor keys do not function in this screen mode.

Type SCREEN 3 and press RETURN. You can see you have entered graphics mode, because there is no longer a flashing cursor. It has changed to an underline character, which is the same as the insert cursor in the other screen modes. Thus the only indication of the PX-8 being in the insert mode is that the INS LED above the keyboard is lit.

As an example of the use of the graphics screen mode, the following command will draw a line from the top left-hand corner to the bottom right.

LINE (Ø, Ø) – (48Ø, 64)

2.15 Input/Output Device Support

PX-8 BASIC supports data input and output (I/O) to and from a variety of peripheral devices. These include random access devices such as external floppy disk drives and RAM disk (a user-specified area in memory which is used in the same manner as an external disk drive), and sequential access devices such as the RS-232C interface, printer and LCD screen.

Devices which have full random access capability allow the records of files to be read or written in any order. All external storage devices used with the PX-8 have some degree of random access capability.

Sequential access devices are devices in which each item of data in a set is input from or output to the device in the order in which it occurs in that set. With PX-8 BASIC, sequential file organization may be used for storage of information in external access devices such as floppy disk drives, or for input/output of information when I/O devices such as the keyboard and RS-232C interface are handled as device files.

2.15.1 Random access devices

Random access devices are devices which can be open in either the random ("R") or sequential ("I" or "O") modes. Random access devices supported by PX-8 BASIC include external floppy disk drives, RAM disk, and the PX-8's built-in microcassette drive. Collectively, these are referred to as disk devices.

Disk devices can be classified into three categories (types I, II, and III) as follows.

(I) Type I disk devices are external storage devices which can be both read and written to on a random access basis. Devices included in this category are RAM disk and external floppy disk drives. All disk I/O statements and functions can be used with type I devices.

(II) Type II disk devices are devices which can be read (but not written to) on a random access basis. Devices included in this category are ROM capsules in ROM socket 1 and 2. Only read statements/functions can be used with this type of device.

(III) Type III disk devices are devices which can both be read and written, but for which random access support is limited. At present, the only device which is included in this category is the PX-8's built-in microcassette drive.

With this type of device, the GET and PUT statements cannot be used concurrently when a file is opened in the random access mode. Further, record numbers must be used in sequence each time the GET or PUT statement is executed. (See the explanations of the GET and PUT statements in Chapter 4.)

When the PX-8's power is first turned on, the device names of the random access devices are as shown below. The name assignments can be changed with the CONFIG command of CP/M; see the PX-8 User's Manual for procedures for doing this. These device names are included in the file descriptor with the file name as described in section 2.12.

RAM disk	A: (in main memory or optional RAM disk unit)
ROM capsule 1	B:
ROM capsule 2	C:
Floppy disk drives	D:, E:, F:, G:
Microcassette drive	H:

2.15.2 Sequential access devices

Sequential access devices are devices which can be open as files for input (the "I" mode) and/or output (the "O" mode). Sequential access devices supported by PX-8 BASIC and the modes in which they can be opened are as follows.

KYBD:	Keyboard, input only
SCRN:	LCD screen, output only
LPTØ:	Printer, output only
COMØ:	RS-232C interface, both input and output

Note that the colon is a part of each of these device names, and must be specified in the file descriptor whenever one of these devices is prepared for input/output by execution of an OPEN statement. However, no file name is specified following the device name when one of these devices is opened as a file.

Statements and functions which can be used with sequential access devices are as shown in section 2.15.7 below.

NOTE:
The format for file specification with the RS-232C interface is slightly different than that of other sequential access devices. See Chapter 6 for details.

2.15.3 Speaker

PX-8 BASIC also supports output to a speaker (either the built-in speaker or an external speaker connected to the SP OUT jack on the back of the PX-8). No device name is assigned to this device; however, output control is possible using the BEEP and SOUND statements.

The speaker can also be used to check for the presence of recorded signals on the tape. This is done by executing the WIND ON statement. (See the explanation of the WIND statement in Chapter 4.)

2.15.4 The analog converter

On the rear of the PX-8 is a socket marked A/D IN. This is the analog-to-digital converter. It takes an analog voltage and converts it into a number. This number is proportional to the voltage. The analog-to-digital converter is not supported by BASIC, but it can be used if a short machine code subroutine is added to a BASIC program. This machine code subroutine is required to make the necessary BIOS call to return the digital value corresponding to the analog voltage. An example of its use is given in the User's Manual, where the voltage across a variable resistor is determined. This is then used as a paddle in a simple game. The program shows how the machine code routine to read the A/D port can be used with a BASIC program.

2.15.5 The bar code reader

Many products are marked with a machine readable coded series of bars. This allows a numerical value to be read simply by moving a bar code wand across the bars. A socket for insertion of the wand can be found on the rear of the PX-8 marked BRCD. Further details are included in the User's Manual.
Examples of the use of a bar code reader include identifying suitably coded products, then using the information in a program to count the number of different items. Special software is required to perform the task of reading the data. Such software must be written in machine code, but can be linked to BASIC. Use of the bar code reader requires the purchase of a separate software package, and details of how to use the software with BASIC are included with the package. Please consult your dealer for more information.

2.15.6 Other devices

Communication with a number of other devices can be achieved if they have an RS-232C serial port. Besides transmitting data of a conventional nature (e.g. text sent over a modem or acoustic coupler) data can also be sent to control other equipment. Please consult your dealer for further information.

2.15.7 Commands, statements, and functions usable with I/O devices

Commands, statements, and functions which can be used with the various I/O devices are as indicated in the table below.

Device	KYBD	SCRN	LPT0	COM0	Disk-I	Disk-II	Disk-III
CLOSE	○	○	○	○	○	○	○
DSKF	×	×	×	×	○	○	†2
EOF	—	×	×	○	○	○	○
GET	×	×	×	×	○	○	†3
INPUT #	○	×	×	○	○	○	○
INPUT$	○	×	×	○	○	○	○
LINE INPUT #	○	×	×	○	○	○	○
LIST	×	○	○	○	○	×	○
LOAD	○	×	×	○	○	○	○
LOC	—	×	×	○	○	○	○
LOF	—	×	×	○	○	○	○
OPEN "I"	○	×	×	○	○	○	†3
OPEN "O"	×	○	○	○	○	×	†3
OPEN "R"	×	×	×	×	○	○	†3
POS	×	○	○	○	○	○	○
PRINT #	×	○	○	○	○	×	○
PRINT USING #	×	○	○	○	○	×	○
PUT	×	×	×	×	○	×	†3
SAVE	×	†1	†1	†1	○	×	○
WIDTH	×	×	○	○	×	×	×
WRITE	×	○	○	○	○	×	○

Disk-I RAM disk, floppy disk drives
Disk-II ROM capsules
Disk-III Microcassette drive

○: Usable ×: Not usable —: Meaningless

†1 Output is in ASCII format even if binary save or protect save is specified.
†2 Value returned is not trustworthy due to the nature of cassette tape.
†3 Use is subject to restrictions.

2.16 Error Messages

Error messages are displayed when errors are detected during execution of BASIC commands, statements or functions. If errors occur during program execution, execution stops and BASIC returns to the command level. However, it is possible to prevent this by including error processing routines which use the ON ERROR statement and ERR and ERL functions in programs; see the next section and corresponding explanations in Chapter 4 for details.

A complete list of the BASIC error codes, error messages, and causes of errors is shown in Appendix A.

2.17 Error Processing Routines

When it is possible to anticipate that a certain line of a program will result in an error or that an error of a particular type will occur, programs can be designed to include routines which are referred to as error traps, or error processing routines. The purpose of an error processing routine is either to evaluate and correct an error or to allow the user of the program to input corrective data; afterwards, the error processing routine either terminates program execution or causes it to resume at a particular point, depending on what conditions are written into the routine.

The ON ERROR GOTO <line no.> statement must be executed in order to define the starting line number of the error processing routine. After execution of this statement, occurrence of any error at any line in the program will cause execution to jump immediately to the line number specified following GOTO. Afterwards, the ERR and ERL functions can be used to evaluate the type of error and the point at which it occurred in the program, and the RESUME statement can be used to transfer execution out of the routine and back to a specific point in the main program.

Errors can also be simulated using the ERROR statement.

See the explanations of the ERROR, ON ERROR GOTO, and RESUME statements and the ERR/ERL functions in Chapter 4 for further information.

Chapter 3

ENTERING BASIC WITH EXTENDED FORMAT COMMANDS

It is possible to enter BASIC with various commands appended which can set up the number of files allowed, upper memory limit etc. This chapter summarises these options.
The full syntax of the BASIC command is as follows:

BASIC [<filename>][/F:<no. of files>][/M:<upper memory limit>] [/S:<maximum record size>][/P:<program area no.>] [/R:<program area no.>]

All command operands indicated in brackets ([ ]) above are optional; the functions of each of the operands are as described below. The brackets are inserted to show the separation of the options and should not be typed in.

(1) BASIC <filename>

Example **BASIC TEST1**

When the name of a BASIC program file is specified following BASIC, that program is loaded and executed upon completion of the BASIC command in the same manner as if RUN "<filename>" had been entered immediately after start-up. If no file name extension is specified, .BAS is assumed. If neither the /R: nor /P: options are specified, BASIC starts operation using program area 1. See section 2.12 for details on <filename>. The above example would enter BASIC and proceed to RUN the program which was named TEST1 on the currently logged in disk drive. It would be placed in program area 1.

(2) BASIC /F:<no. of files>

Example **BASIC /F:5**

The /F:<no. of files> option sets the number of files which can be open simultaneously. In the example the number would be 5. If this operand is omitted, the maximum number of files which can be open simultaneously is set to 3 (the system default value). The maximum value which can be specified in <no. of files> is 15. In this statement "files" refers to data files, either for communication or saving and loading to a disk drive. For example

OPEN "O", #3, "FILENAME"

If a file number is used which is greater than the number specified in the option, a

Bad file number in <line number>

error will occur.

(3) BASIC /M:<upper memory limit>

Example **BASIC /M:&HC000**

The /M:<upper memory limit> option specifies the highest address in RAM which can be used as program or variable area by the BASIC interpreter. The value specified for <upper memory limit> must be smaller than the starting address of the basic disk operating system (BDOS). The memory area starting at the address from <upper memory limit> to the beginning of BDOS can then be used for storage of machine language programs. Naturally, this reduces the amount of memory which can be used for variables or storage of BASIC programs. However, it must be noted that the BDOS starting address will vary according to the number of bytes of memory reserved for use as the RAM disk. The starting address of BDOS can be found by looking in page zero. Locations 5, 6 and 7 contain a jump to BDOS. Therefore locations 6 and 7 contain the starting address of BDOS, in the order LSB, MSB. To obtain this from BASIC use:

PEEK(6) + PEEK(7) * 256

In the example above, the upper memory limit is specified as a hexadecimal number; however, it can also be specified as a decimal number. Full details of this are described in the section on the CLEAR command in Chapter 4.

(4) BASIC /S:<maximum record length>

Example **BASIC /S:256**

The /S:<maximum record length> option sets the maximum record length which can be used with random access files to the value specified in < >; the maximum record length can be specified in decimal, hexadecimal (&H) or octal (&O) notation. When an OPEN "R" statement is executed after starting BASIC, the record size specified in that statement cannot be larger than the value specified with this option. If this option is not specified when the BASIC command is executed, the maximum record size is set to 128 bytes.

(5) BASIC /P:<program area no.>

Example **BASIC /P:3**

The /P:<program area no.> option specifies the program area which is selected at the time BASIC is started and automatically logs into that area. The value of <program area no.> must be specified as a number from 1 to 5.

(6) BASIC /R:<program area no.>

Example **BASIC /R:3**

As with the /P: option, the /R:<program area no.> option starts BASIC and selects and logs in the specified <program area no.>; in addition, this option immediately executes any BASIC program which is present in the specified program area. As with the /P: option, <program area no.> must be specified as a number from 1 to 5.

If both the /P: and /R: options are omitted, the BASIC program menu described in section 1.3 is displayed when BASIC is started.

If any errors are made while entering the BASIC command, an error messsge is displayed and the MENU screen or system prompt is redisplayed (depending on whether or not the MENU screen function is turned on). This also occurs if sufficient memory is not available for the BASIC working area (either because the upper memory limit or the starting address of BDOS is too low).