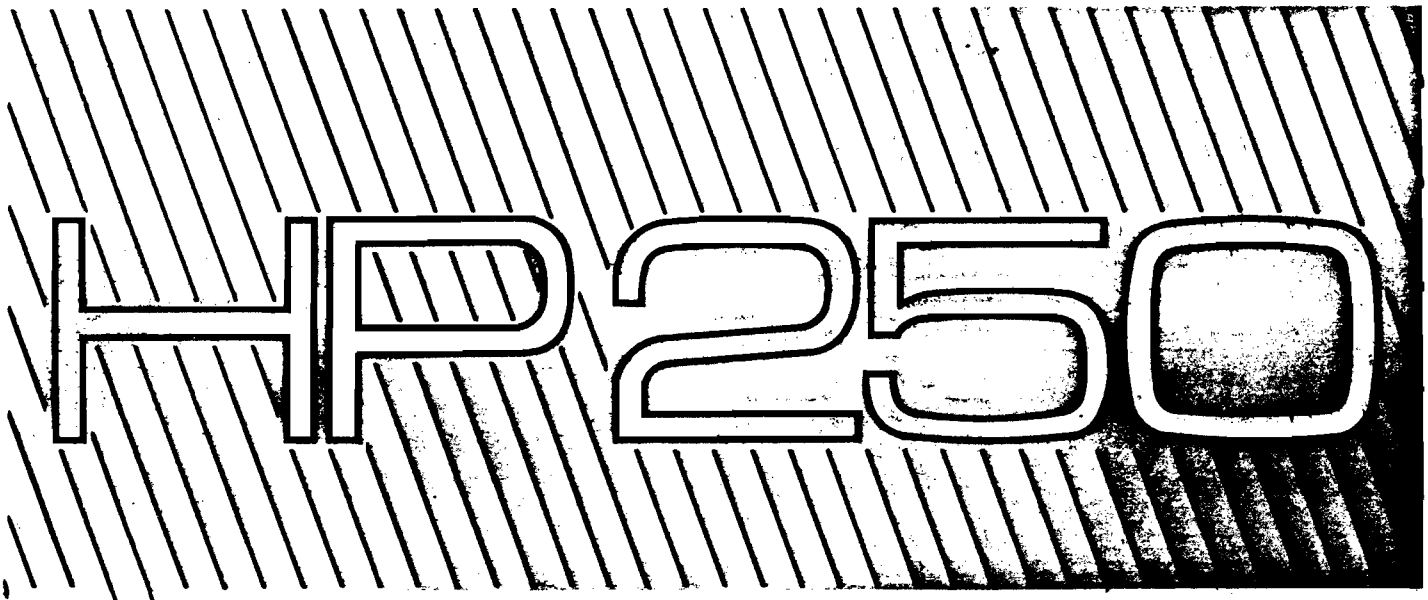
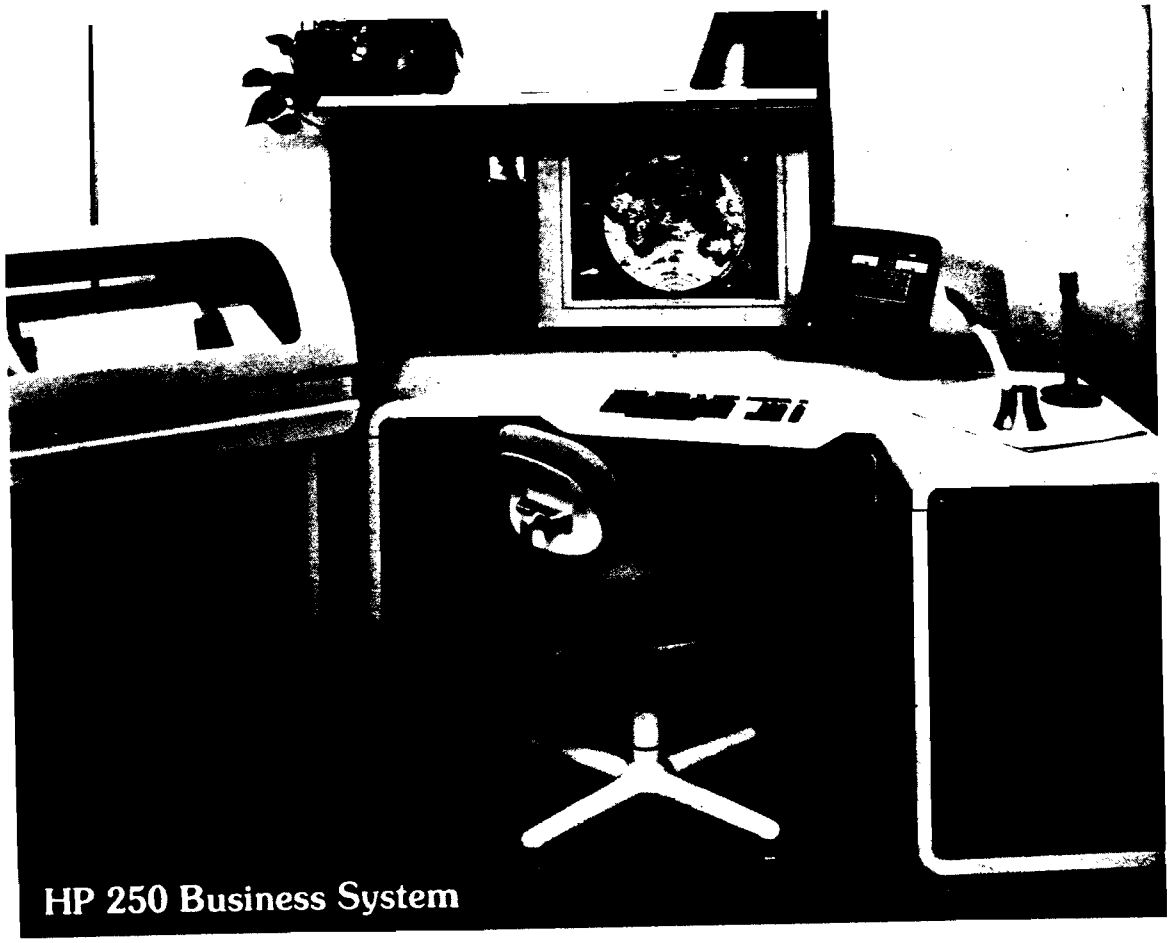


Pre-study Workbook

HP 250 Customer Training
Pre-study Workbook



Hewlett-Packard
Fort Collins Division
19447 Pruneridge Ave., Cupertino, California 95014 U.S.A.
(For World-wide Sales and Service Offices see back of manual.)

Preface

The purpose of this document is to prepare you for your upcoming HP250 Customer Course. Two distinct subjects are presented – the BASIC Programming Language and a conceptual overview of a Data Base Management System.

The chapter concerning Data Base Management should be read as an introduction to provide you with concepts. It represents an overview of data base structure as well as traditional file structures. This document is also helpful in familiarizing you with data base terminology.

The BASIC section should be used in conjunction with the BASIC Programming Manual. The workbook does not provide syntax as does the manual. It is recommended that you read and study the programming manual first, then complete the exercises in the Workbook. Allow yourself one to two weeks to study the manual and write the solutions to the exercises. The HP250 Customer Course does not address BASIC in a syntactic sense. However, important statements and commands are reviewed via a lecture. In addition, lab time is provided during class for you to enter and de-bug the program solutions you wrote.

Please take the time to study the recommended materials. A rigorous week of training has been planned for you, and it is our intent to optimize your time and learning experience.

BASIC/250

Objectives

The objective of this chapter is to provide you with an overview of the BASIC statements and convey programming concepts through lab exercises. We strongly recommend reading and studying the BASIC programming manual prior to examining this workbook. You should study the Programming Manual first, then read and work through the exercises in this workbook. The lab exercises should be handwritten before attending the HP250 Customer Course. There will be lab time provided during the class so that you can enter the programs you have written.

The statements are described in this workbook in summary form. Beside each statement is listed the page number of related information in the BASIC Programming Manual. In the left margin are two columns which designate a statement as programmable or keyboard executable. If a statement is programmable it can be executed in a program. Therefore, it must have a line number. Some statements are not programmable and are only executable from the keyboard. We often refer to these special statements as commands. Some statements are both programmable and keyboard executable. For example, READ is a programmable statement, RUN is an executable statement, and STOP is both! When a statement is keyboard executable, it must be typed and then entered by pressing ⇧ .

Overview of HP250 BASIC

The HP250 BASIC is an enhanced version including many functional statements typical of other commercial languages. The emphasis of this workbook is on the function of statements, rather than syntax. HP250 BASIC includes several powerful statements and concepts that HP considers major contributions to the language. A brief description of these enhancements follows:

The CURSOR statement is included in the HP250 BASIC. This statement defines the display enhancements to be used with output to the CRT. The cursor can also be positioned anywhere on the screen through the CURSOR statement.

In order to increase user interactiveness with programs, the Special Function Keys can be defined. In order to define these twenty-four keys, the ON KEY # statement is used. These keys can be pressed by a user, and a defined program branch is executed. There are 16 Special Function Keys (SFK's) on the keyboard (actually only eight keys, but each able to be doubly defined) and eight on the CRT. The SFK's can be defined as typing or programming aids, or as mentioned previously, program interrupts.

Variable names can be 1 through 15 characters in length, allowing names to be more descriptive. Consequently, programs become more self-documenting. Line labels are also helpful in making execution transfers within a program more obvious and well-documented. Line labels follow the same rules as variable names for length and character type.

The HP250 BASIC has string capabilities with numerous string functions associated with them. String variable implementation makes data manipulation easier.

Subprograms are also incorporated into the BASIC language. The use of subprograms provides for modularization of programs, which in turn, leads to better organized code.

The PRINT LABEL statement allows a medium to be assigned a name and accessed through this name. This feature is particularly useful since the medium, for example, can be accessed independently of the device location.

It is the purpose of this workbook to illustrate these BASIC enhancements through organized labs. We are now ready to begin ...

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

System Commands

There are several commands associated with manipulating the system. These commands deal with program execution, purging of files, selecting an output device, and defining Special Function Keys as programming or typing aids. In order to execute any of the following commands, type the command (with proper parameters) and press ⇧.

Program Editing

The commands below are used to enter, retrieve, edit, and list lines. These commands are easy to use and the command name is indicative of its function.

Programmable	Keyboard executable	
	X	AUTO automatically numbers program lines. (1-14)
	X	REN automatically renumbers program lines. (1-14)
X	X	DEL deletes program lines from memory. (1-14)
	X	FETCH retrieves and displays the specified line. (1-15)
	X	LIST lists program lines to the system printer (which is the specified output device or if not specified, the default device is the CRT). (1-15)
	X	SPACE DEPENDENT allows typing in uppercase mode with special note to spacing. (1-27)
	X	SPACE INDEPENDENT allows typing with no regard to spacing but correct uppercase or lower-case characters must be noted. (1-28)

Program Execution

The commands that follow are used to execute a program, stop and then continue execution, erase memory, prevent lines from being listed, and to declare an output device. They are described below.

Programmable	Keyboard executable	
	X	RUN executes a program. (1-13)
	X	CONT continues execution after a PAUSE or HALT. (1-13)
	X	SCRATCH erases memory. (1-26)
X	X	STOP halts program execution and resets program line counter. (1-29)
	X	SECURE prevents program or program lines from being listed. (1-26)
X	X	SYSTEM PRINTER IS defines output device for LIST and CAT. (7-4)
X	X	PRINTER IS defines output device for PRINT, PRINT USING statements. (7-3)
X	X	PRINT ALL IS defines output device for all user/system interaction. (7-4)

Typing/Programming Aids

The next four commands refer to assigning, listing, saving, and retrieving the SFK definitions assigned by a programmer. These SFK's can be defined as programming or typing aids.

Programmable	Keyboard executable	
	X	EDIT KEY # provides for SFK definition. (1-30)
	X	LIST KEY # lists definitions of any or all SFK's. (1-31)
	X	STORE KEY stores SFK definitions in a file. (6-52)
	X	LOAD KEY retrieves SFK definitions from a file. (6-52)

Recommendations for entering programs:

- Begin line numbering with 1000 and by increments of 10. This technique ensures that the left margin of the program is uniform.
- Indent "subsections" of your program more than other lines. These sections become more easily identifiable.
- Use comments and remarks generously. Your program becomes better documented and easier to follow.
- Indent all lines at least 15 spaces so that line labels can be inserted at any time. If a line label was inadvertantly omitted or the need arises for a line label to be inserted, the above technique makes it easy.

Exercise 1A

Assign the following definitions to the SFK's 9 – 14 (on the keyboard)

```
# 9  PRINT
#10  REM
#11  DISP
#12  INPUT
#13          (15 blanks)
#14          (20 blanks)
```

Use these typing aids to help you enter the program in the next exercise. You should store your key definitions in a file called "Keys_1". Verify that "Keys_1" is on your disc.

Exercise 1B

1. Enter this program into memory:

```
10   REM   THIS IS EXERCISE 1B...EMPLOYEE PHONES
20   DIM Employee$(10,3)[20]
30   INTEGER I
40   INPUT "ENTER THE NUMBER OF EMPLOYEES (<10):";I
50   FOR J=1 TO I
60   INPUT "EMPLOYEE  NAME:";Employee$(J,1)
70   INPUT "EXTENSION:";Employee$(J,2)
80   INPUT "DEPARTMENT #:";Employee$(J,3)
90   PRINT TAB(10);Employee$(J,1);TAB(40);Employee$(J,2)
;TAB(70);Employee$(J,3)
100  NEXT J
110  END
```

2. Get a listing of the program both on the display and the printer.
3. Run the program and enter this sample data for 3 employees:

Bill Cummings	2130	MKTING
Rex James	2305	LAB
Tim Eckerman	2479	PROD

4. Secure lines 60 thru 90.
5. Get a 3rd listing.

Programming Statements

The following programming statements are categorized according to function. For more information regarding each statement, **refer to the page number of the BASIC Programming Manual, which is provided in parentheses.**

Declaring Variables

A programmer should declare the variables used in a program. Either the variable precision can be declared (REAL, INTEGER, or SHORT) and/or the dimensions of any string or array variable. The precision specifies the maximum number of significant digits a variable value may contain. By default a numeric variable assumes REAL precision. The dimensioning of string variables, that is, declaring the maximum length, and designating the number of dimensions of an array, is accomplished through a DIM statement. A new working size of an already dimensioned array can be established by REDIM.

Programmable	Keyboard executable	
X		SHORT declares a variable value with 6 significant digits and an exponent of −63 thru 63. (2-18, 2-5)
X		INTEGER declares a variable value from −32768 thru 32767. (2-18, 2-5)
X		REAL declares a variable value with 12 significant digits with an exponent of −99 thru 99. (2-19, 2-5)
X		COM declares variables to be shared by more than one program or subprogram (or function). (2-19)
X		DIM declares dimensions on arrays, maximum length for strings, etc. (2-18)
X		REDIM defines a new working size for an array. (2-21)
X		OPTION BASE defines the lower bound of the array subscripts to be either 1 or 0 (default is 0). (2-17)

Entering Data

Variables can be assigned values either from the keyboard or from within a program. Keyboard entry is provided by the INPUT, LINPUT, ENTER, LENTER, and EDIT statements. READ, DATA, and RESTORE are used in combination to assign values from within a program. The keyboard entry is of particular use since input from special locations or lines of the CRT are allowed.

Programmable	Keyboard executable	
X		INPUT is used to enter values from the keyboard during program execution. (2-25)
X		LINPUT (LINE INPUT) is used to enter a line of any characters from the keyboard. (2-26)
X		EDIT allows a string to be modified and re-entered. (2-27)
X		ENTER (used primarily with FORMS) allows a program to read data from the display. (2-28)
X		LENTER (LINE ENTER) allows a program to automatically read a specified line on the display. (2-29)
X		READ/DATA/RESTORE allows assignment of values to variables from within the program. (2-22)

Exercise 2A

Modify program in exercise #1B so that each line of input can be checked for errors and edited, if necessary, by the program user (i.e., use EDIT).

Looping

The two statements – FOR and NEXT – are used to establish controlled looping within a program. The FOR statement defines the number of times the loop is to be performed, and the NEXT statement increments and tests the loop counter.

Programmable	Keyboard executable	
X		FOR/NEXT/STEP defines controlled looping. (4-6)

Transfers

Unconditional, conditional, and computed transfers are allowed within programs. Transfers are well-documented when line labels are used. Of special note is the ON KEY # statement, which defines a transfer to be performed when a particular SFK is pressed. The use of the ON KEY # statement increases user interaction with programs.

X		GOTO transfers execution unconditionally. (4-2)
X		IF/THEN defines conditional branching or statement execution. (4-4)
X		GOSUB transfers execution to a subroutine. (4-11)
X		RETURN returns control from a subroutine to the line immediately following the calling statement. (4-11, 5-5)
X		ON/GOTO provides a transfer to one of several lines dependent on the value of a variable. (4-3)
X		ON/GOSUB defines a transfer to a subroutine dependent on the value of a variable. (4-12)
X	X	ON KEY # defines a transfer to occur when a certain SFK is pressed. (4-14)
X	X	OFF KEY # deactivates the specified ON KEY # condition. (4-18)

Programmable	Keyboard executable	
X		ON HALT causes a branch to occur when the HALT key is pressed. (4-22)
X		OFF HALT deactivates the ON HALT condition. (4-22)
X		ON ERROR causes a defined branch to occur when an error results. (4-20)
X		OFF ERROR deactivates the ON ERROR condition. (4-20)

Output

Information can be output either to the CRT or to the printer for viewing. The output device (CRT or printer) can be specified for output through these statements: SYSTEM PRINTER IS, PRINTER IS, and PRINT ALL IS. The DISP, and DISP USING statements direct output strictly to the display. PRINT and PRINT USING direct output to the device specified. CURSOR is used to enhance output only directed to the CRT.

X	X	DISP causes items to be displayed on CRT. (7-7)
X	X	PRINT causes items to be printed on the specified output device. (7-24)
X	X	CURSOR positions cursor on display and defines the enhancements for output. (7-14)
X		DISP USING formats output to CRT. (7-27)
X		PRINT USING formats output to standard output device. (7-26)
X		IMAGE defines specifications for formatted output. (7-27)
X	X	SYSTEM PRINTER IS defines output device for CAT, LIST. (7-4)
X	X	PRINTER IS defines output device for PRINT and PRINT USING statements. (7-3)
X	X	PRINT ALL IS defines the output device for output of all user/system interaction. (7-4)

Exercise 2B

Write a short program that inputs up to 25 numbers from the keyboard. The program user can select to have the MAXIMUM, MINIMUM or AVERAGE of the set calculated, as well as a list of the original 25 numbers. Use ON/GOSUB or ON KEY # to accomplish the correct transfer. Use DISP USING to format all output to the CRT.

Program Termination or Suspension

There are several statements used to either halt program execution or merely suspend it. END and STOP are functionally equivalent; however, END is used for documentation purposes while STOP most frequently separates a main program from its subprograms.

Suspension of program execution is accomplished through PAUSE, INPUT, and WAIT. PAUSE is used mainly to modify a program during execution. INPUT activates an input mode used in the HP250/FORMS software. WAIT is used most often to allow a user to make a softkey selection.

Programmable	Keyboard executable	
X		END terminates program execution, resets program line counter. (1-29)
X	X	STOP terminates program execution, resets program line counter. (1-29)
X		PAUSE suspends program execution until CONT is typed. (1-29)
X		INPUT (with no parameters) suspends program execution until data is entered into memory. (2-25)
X		WAIT delays program execution either the specified # of milliseconds or if no time is noted, until an SFK is pressed. (4-15)

Exercise 2C

Write a short program that allows an operator to add program lines to it while it is executing (i.e., use PAUSE). For example, PRINT or DISP statements could be inserted in the program for debugging purposes.

Functions

There are several built-in functions that are useful to a programmer. Those functions can be categorized into numeric, string, mass memory, and error documentation functions. All of these groups except mass memory are discussed below.

Numeric

The following functions have arguments which are numeric. That is, each function acts on a numeric expression only. And consequently, the result of each function is also a numeric expression.

Programmable	Keyboard executable	
X	X	ABS numeric expression returns with absolute value of the numeric expression. (3-8)
X	X	DROUND (numeric expression, number of significant digits) returns the numeric expression rounded to the specified number of significant digits. (3-8)
X	X	EXP numeric expression returns the value of Napierian e (= 2.718) raised to the power of the computed expression. (3-8)
X	X	FRACT numeric expression returns the fractional part of the evaluated expression and is defined by: argument − INT (argument). (3-8)
X	X	INT numeric expression returns the greatest integer which is less than or equal to the evaluated expression. (3-8)
X	X	LGT numeric expression returns the common logarithm (base 10) of a positive numeric expression. (3-8)
X	X	LOG numeric expression returns the natural logarithm (base e) of a positive expression. (3-8)
X	X	MAX (list) returns the largest value in the list of numeric expressions. (3-8)

Programmable	Keyboard executable	
X	X	MIN (list) returns the smallest value in the list of numeric expressions. (3-8)
X	X	PI returns the value of PI. (3-8)
X	X	PROUND (numeric expression, power-of-ten position) returns the numeric expression rounded to the specified power-of-ten position. (3-8)
X	X	RND generates a pseudo-random number greater than or equal to 0 but less than 1. RANDOMIZE re-evaluates the random number seed. (3-8)
X	X	SQR numeric expression returns the square root of a non-negative expression. (3-8)
X	X	DIV (numeric expression/numeric expression) returns the integer value of the calculated division. (B-2)
X	X	numeric expression MOD numeric expression returns the integer remainder of the division of the first expression by the second. (B-2)

String Functions

The following functions are associated with string variables. The arguments of the various functions are not exclusively string in nature, nor does each function always return a string.

X	X	CHR$ (X) returns the ASCII character-equivalent of the numeric expression X. The value of X must be between 0 and 255. (3-14)
X	X	RPT$ (S$,X) repeats S$ for X times. (3-14)
X	X	TRIM$ (S$) returns a string which is equal to S$ with all leading and trailing blanks stripped off. (3-14)

Programmable	Keyboard executable	
X	X	VAL$ (X) converts the value of the numeric expression (X) to its corresponding string of ASCII digits. (3-14)
X	X	LWC$ (S$) returns a string with all lowercase characters. (3-14)
X	X	UPC$ (S$) returns a string with all uppercase characters. (3-14)
X	X	LEN (S$) returns the number of ASCII characters in the string S$. (3-14)
X	X	NUM (S$) returns a numeric value between 0 and 255 corresponding to the first character of the string S$. (3-14)
X	X	POS (S$,T$) searches the string S$ for the first occurrence of the string T$. Returns the starting position (index) if found; otherwise returns 0. (3-14)
X	X	VAL (S$) returns the numeric equivalent of the string S$, which must be a string of ASCII digits. S$ may include a decimal point. (3-14)

Error Documentation

The following functions are associated with documenting errors so that error recovery can be provided during program execution.

X	X	ERRL returns the line number in which the most recent program execution error occurred. (4-20)
X	X	ERRM$ returns the most recent error message encountered in the program. (4-20)
X	X	ERRN returns the number of the most recent program error. (4-20)

Exercise 2G

1. Write a program to compute and print all multiples of 7 and 11 between 0 and 250. Use the INT function.
2. Write a program that reads in 5 sample sets of customer data (name, address, phone, etc.) where all data is contained in strings. Test which customers have Colorado addresses, and print out a list of those customers.

    1. R2-J School District, 711 Lincoln, Loveland, Colorado 80537
    2. B.A.C. Corporation, 1207 Main, Chicago, Illinois 60618
    3. Compudec Inc., Box 1983, Atlanta, Georgia 40739
    4. Barnes Hospital Supply, Barnes Medical Center, St. Louis, Missouri 70439
    5. Calculator Sales, 4273 E. Evans, Denver, Colorado 80658

Writing To/Reading From A File

Concepts

All data files must be created, assigned a #, and opened before they can be accessed. This is accomplished via the CREATE and ASSIGN statements. A CREATE statement establishes a file of the specified size on the storage medium. When an ASSIGN statement is executed, an internal table of file numbers is set up for use by the PRINT # and READ # statements. Each opened data file has a record pointer which is automatically maintained. This pointer specifies the record being accessed in the file. A word pointer is also maintained for each current record and points to the first word of the next data item to be accessed in that record.

Data can be retrieved from a data file in three ways: serial access, direct access and direct word access.

Serial access to a file is used to store or retrieve data (one item after another) without regard to logical record bounds. Pointers move serially through the file as data is stored or retrieved.

Direct access is used to retrieve data from individual logical records. All data is stored at or retrieved from the beginning of the specified record, and not at the current position of the pointer.

Direct word access enables access to a data file at any given word within the specified record. The word pointer must be specified in the syntax of the PRINT # or READ #, and must be an integer from 1 thru 129.

Although a file is automatically closed when a STOP or END is executed, the ASSIGN statement can also be used to close a file. When a closed file is accessed, an error will result.

The following collection of statements is related to creating, opening, closing, reading from and writing to files, as well as trapping the EOF mark.

Programmable	Keyboard executable	
X	X	CREATE establishes a data file on the storage medium. (6-23)
X	X	ASSIGN opens/closes a data file. (6-25, 6-40)
X		PRINT # records values in a data file. (6-26, 6-32, 6-35)
X		READ # retrieves values from a data file. (6-28, 6-34, 6-37)
X		ON END # defines a branch to occur when an EOF mark is detected (or an EOR during a random PRINT #) (6-46)
X		OFF END # cancels a previous ON END # statement. (6-47)

Exercise 2D

Write a program that creates a data file called EMPLOY. Open that file and store up to 10 employee names and ID's there. Retrieve that data and output it in a list. Close the file.

In order to re-run that program, you will have to PURGE the file before re-CREATEing. Alternatively, the program can detect whether the file is already created by using ON ERROR or the RETURN variable in the ASSIGN statement.

Use this sample data:

1.	Mabel Blue	72031
2.	Rex Barlow	52847
3.	Mike Jones	88075
4.	Mary Brownstone	94723
5.	Clayton Munson	51582
6.	Jeff Bernard	24531
7.	Karen Klein	47836
8.	John Hammel	14739
9.	Jack Thomson	52818
10.	Mercella Roe	17225

Mass Storage Statements

After a program or data has been entered into memory, that information can be stored in a file on a disc. Files can be created for various uses, and depending on how that file is created, determines the file type.

The most commonly used types of files are:

Program files
Data files

However, there are files used in this course related to FORMS and data sets for IMAGE. A file consists of one or more records, which each contains 256 bytes of storage. There are two types of records – physical and logical. Physical records are established when the medium is initialized. Every file starts at the beginning of a physical record. A logical record is established thru a CREATE statement and can have 1 thru 65534 bytes. A logical record is the smallest unit of storage which is directly accessible.

When a file is written to the disc via a STORE or SAVE statement, as many 256-byte physical records are used as is needed. When CREATE is used to define a data file, the number of logical records for that file can be specified. However, the first logical record is positioned at the beginning of a physical record.

Each storage medium, i.e. a disc, has a directory which maintains information regarding file name, type, length, location and loading information. You may view a catalog of similar information by executing a CAT on the medium in question.

When the system is powered on or when SCRATCH A is executed, the mass storage device from which the operating system was loaded is automatically specified as the standard storage device. This is the device to which all file storage operations are directed if no other device is specified. The standard device can be changed by using the MSI statement.

The flexible disc drive doors may be locked (as during updates on discs) by using the DOOR LOCK statement. The door can be unlocked via the DOOR UNLOCK statement. A label may be assigned to a volume (medium) through the PRINT LABEL statement, allowing access to the media independent of the drive unit. Files can be stored and retrieved through several statements. Those statements are discussed below:

Programmable	Keyboard executable	
X	X	CREATE establishes a data file with EOF's. (6-23)
X	X	FCREATE (fast create) establishes a data file without EOF's. (6-23)
X	X	STORE stores program lines in a program file in internally coded format. (6-18)
X	X	LOAD retrieves a program file stored in internally coded format. (6-18)
X	X	SAVE stores a program as string data in a data file. (6-19)
X	X	GET retrieves a data file from medium. (6-20)
X	X	LINK retrieves a file stored by SAVE without erasing the values of variables. (6-21)
X	X	MERGE merges program statements at beginning or end of current program. (6-22)
X	X	RESAVE saves new version of a program under former name. (6-21)
X	X	RENAME saves a program under a new name. (6-52)
X	X	LOAD SUB loads subprograms into memory from disc. (6-19)
X	X	DEL SUB erases specified subprograms from memory. (5-9)
X	X	LOAD BIN loads a binary program into memory. (6-53)
X	X	RE-STORE stores a program into a file created by STORE, STORE KEY, or STORE BIN. (6-19, 6-52)
X	X	PROTECT assigns a protect code to a file to limit access. (6-49)
X	X	PURGE erases a file. (6-41)
X	X	MASS STORAGE IS establishes the default mass storage device. (6-11)

Programmable	Keyboard executable	
	X	CAT lists all files currently on specified medium. (6-12)
X	X	assigns a label to a medium. (6-16)
X	X	reads the label, if it exists, from the specified medium. (6-16)
X	X	locks specified flexible disc drive. (6-9)
X	X	unlocks specified disc drive. (6-10)

Mass Memory Functions

These functions return information concerning available media space, record pointers, record size, string data lengths, and data type. The functions are particularly useful when accessing data files.

X	X	AVAIL returns the total number of available records on the medium on the current standard mass storage device. (6-45)
X	X	HOLE returns the largest number of available contiguous free records on the current standard mass storage device. (6-44)
X	X	REC ([−] file number) returns the current position of the record pointer for the specifed mass storage file. Specifying a negative file number returns the physical record number for a specified file. (6-44)
X	X	SIZE ([−] file number) returns the size of the specified file, in logical records. Specifying a negative file number returns the size of a logical record, in words. (6-44)
X	X	SLEN ([−] file number) returns the number of string characters in the file, beginning at the current word pointer location. (6-44)
X	X	TYP ([−] file number) returns a value which indicates what type of data will be accessed next in the specified file. (6-42)

Exercise 2E

Write a short number guessing game (or a game of your choice) using the RND and INT functions. Declare the mass storage device to be the top disc drive. Use the PRINT LABEL command to assign your name as a label for your disc medium.

1. Store the above program as "GUESS". Write a second program that prints out a simple interest table for loan values of $100 – $1000 and for selected interest rates of 6 ¾% thru 20%. Store the program as "TABLE".
2. Rename "TABLE" by calling it "INTST1".
3. Get a CATalog of the disc contents.
4. Copy "INTST1" to a file called "INTST2".
5. Rename "INTST2" as "INTST3".
6. PURGE "INTST2" (will get error).
7. PURGE "INTST3".
8. Merge file "INTST1" with "GUESS" by appending "GUESS" to the end of "INTST1".
9. Save the new file as "MERGE1".
10. Get a listing of the new program and protect it with a code.
11. Attempt to PURGE that file.

Subprograms/Subroutines

Modular programming lends itself to better organized programs. The use of subprograms provides for this modularity. Subroutines, on the other hand, are used most effectively to reduce repetition of code. Following is a description of those statements associated with subprograms and subroutines.

Programmable	Keyboard executable	
X		CALL calls the desired subprogram and may pass values. (5-7)
X		SUB designates the beginning of the subprogram. (5-7)
X		SUBEXIT transfers control from the subprogram back to the main program. (5-8)
X		SUBEND the documentation end for the subprogram. (5-7)
X		DEF FN specifies a user defined function which returns a single value. (3-17, 5-5)
X		FN END optional last line of a function subroutine. (5-5)
X		DEL FN erases specified user-defined function subprograms from memory. (5-9)
X	X	LOAD SUB loads subprograms into memory from disc. (6-19)
X	X	DEL SUB erases specified subprograms from memory. (5-9)
X		GOSUB transfers control to the line specified. (4-11)
X		ON/GOSUB transfers control to one of the lines specified dependent on a given value. (4-12)
X		RETURN returns control from the subprogram to the line immediately after the calling statement. (4-11, 5-5)

Exercise 2F

Write a program that generates bills for the customer of a small corner grocery. The output should be accomplished through a subprogram. Restrict yourself to this small set of sample data:

1.	John Doe	181 S. College Fort Collins	Amt. due; 58.28
2.	Bill Brown	5870 W. Maple Apt. 7 Fort Collins	Amt. due: 17.37
3.	Joe Hill	RR#2 Fort Collins	Amt. due: 23.75
4.	May Graves	1753 E. Linden Fort Collins	Amt. due: 27.30
5.	Edith Roe	4231 Valleybrook Ln. Fort Collins	Amt. due: 41.92

Utilities

Four very helpful BASIC utilities are the INIT, DUPL, ROUTIL and AUTO START programs. INIT, DUPL, and ROUTIL are BASIC programs and executable with the RUN command. The AUTO START utility is accessible through the CONFIG program.

- RUN "INIT" will run the initialization program. During initialization, the media is tested for defective tracks, physical records are marked, and main and spare file directories are marked. (C-1)
- RUN "DUPL" will run the media duplication program. During duplication of a medium all files are copied from the "source" media to the "destination" medium. The DUPL program is located on the system disc. (C-1)
- RUN "ROUTIL" will run the BASIC language utility that copies and/or purges run-only programs and more importantly, defines a program to be run-only. The ROUTIL is a BASIC program found on the System Disc. (C-1)
- AUTO START program
  The auto start routine allows the system to either display a message or execute a command immediately after the system is loaded at power-on.

  You must enter 7 during the initial CONFIG menu. Once in the AUTO START routine, you can define a command or message to be executed or displayed at power-on. (C-1)

IMAGE/250

Objectives

The purpose of this workbook is to get you thinking about "Data Base Management Concepts". It is designed as a pre-study preparation for your formal HP 250 training.

If you have worked on another machine with a data base management system and feel quite comfortable with the concepts and terminology of data base management, it is suggested that you simply scan this pre-study material to familiarize yourself with the HP 250 Data Base Management System. If, however, data base is new to you, it is suggested that you study this text quite carefully.

What's It All About, Alfie?

You're probably sitting there thinking to yourself, "What in the world is ' data management '? I don't know if I have questions about it since I don't even know WHAT IT'S ALL ABOUT". We hope that when you've finished this section you'll have a good understanding of the problems and challenges of managing data in computer systems and the various methods of dealing with them.

DATA MANAGEMENT consists of the function of managing data stored in a computer system. Currently there are many different terms referring to various aspects of data management – you're probably familiar with such terms as file system, record management services, data base management systems, inquiry programs, etc. Before we examine the details and differences between traditional file structure (organization methods and access methods) and data base management systems, let's quickly review the basic definition and contents of any file.

Files Made Simple

The general purpose of any computer system is to input, process, and output information. A FILE IS A COLLECTION OF INFORMATION OR DATA IDENTIFIED BY A NAME. It is helpful to think of a file in the traditional way that people employed in business and commerce have for many years – as a storage cabinet containing information of various kinds. (See Figure 1.) Instead of a filing cabinet, of course, computers use media such as disc, cards, and tape for storing the information. On any of these media, a file may contain system or user programs, or DATA – alone or in any combination. For instance, a disc file containing a PAYROLL PROGRAM (to be brought into memory and executed) would reference a second file called PAYROLL DATA (wage and salary information). This second file may be stored on the same disc or another disc.

Within a file, all information is organized into units of related data called LOGICAL RECORDS. These records usually are similar in form, purpose, and content. The records in the file can be arranged in almost any order – alphabetically, numerically, chronologically, by subject matter, and so forth. For example, in the payroll file, each logical record could contain the wage and salary data related to a particular employee. There would be one record for each employee, and the records could be arranged in alphabetical order according to the employees' last names. Returning to the cabinet file analogy, the logical record would correspond to a sheet of paper serving as the employee's payroll record, stored in the cabinet. The logical record is the SMALLEST collection of data directly addressed. You should specify its length when you create the file. User programs, however, can also recognize fields or data items within each record. For instance, a payroll record for an employee might contain a field for each of the following items: the employee's name, social security number, marital status, gross pay, tax exemptions, individual deductions, and net pay.

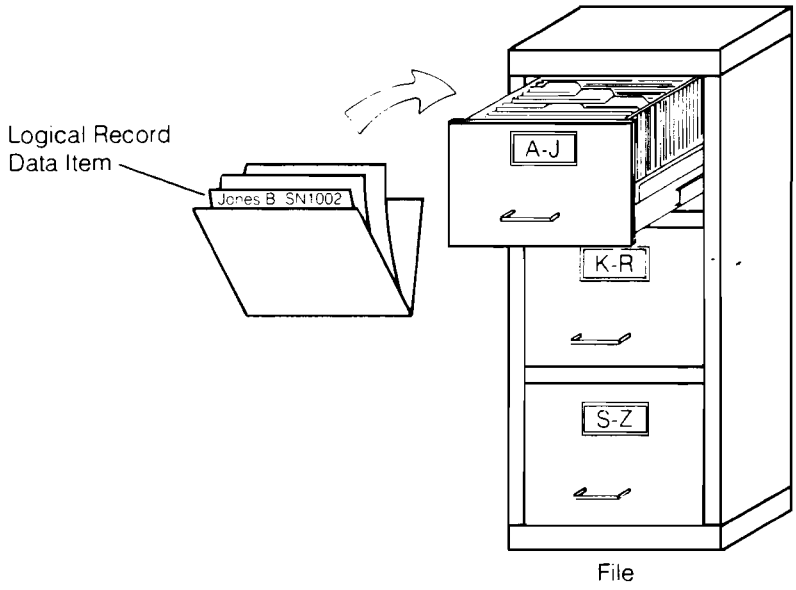


Figure 1. File Analogy

To summarize the relationship of files and logical records:

A FILE is a collection of records treated as a unit and recognized by a name.

A LOGICAL RECORD is a collection of fields treated as a unit residing in a file.

File Organization

File organization refers to the logical construction of the file, i.e., how data within the file can be retrieved. There are three types of file organization – sequential files, relative files, and direct files – which are illustrated on the next page.

Sequential Files

Sequential files are those in which each record must be retrieved in succession. For example, let's consider a file of data concerning a company's payroll. Each week every employee is to be paid. The records corresponding to the employees' payroll information will all be processed, one after another. (See Figure 2.) In this case, a sequentially organized file satisfactorily meets the processing requirements.

	Employee Name	Employee Number	Department	Social Security Number
1	Martin, Joan	01728	Mktg	793
2	Mayer, Bob	35692	Lab	286
3	Kobis, John	27338	Mktg	511
4	Edwards, Rich	47012	Mktg	264
5	Welsch, John	53529	Lab	169
6	Barkley, Bill	19415	Manuf	854
7	Yu, John	23308	Lab	298
8	Colwell, Virginia	32208	Mktg	473

Each record is processed serially in physical order.
Example: Read the next record.

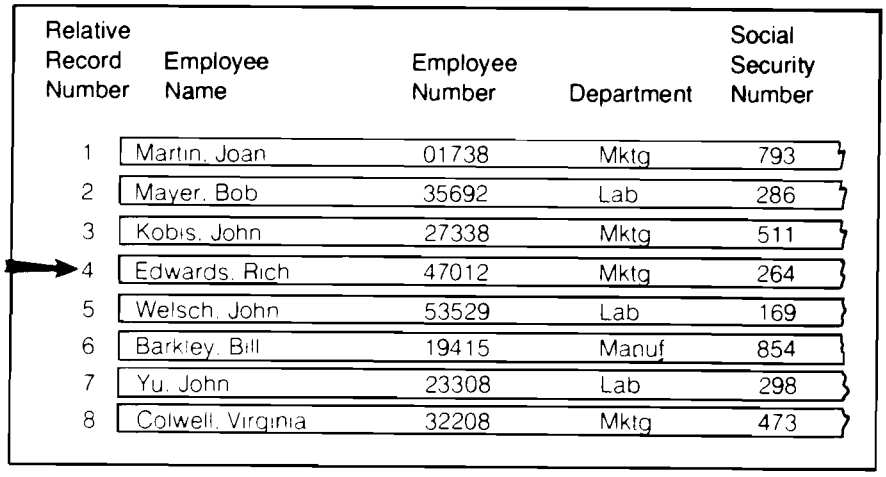
Figure 2. Sequential File

Files stored on magnetic tape are ALWAYS sequential files; to read a particular record requires all preceding records to be read first. Sequential files may exist on other media, such as cards (processed by a card reader), input data from a terminal, discs, and line printers (one output line printed following another).

Note that either to insert a new record or to delete an existing record from a sequential file requires that the entire file be processed and a new file be created with the desired additions or deletions at the appropriate positions in the new file. This is one limitation of using magnetic tape for processing data; discs allow much more flexibility as illustrated in the next two sections.

Relative Files

Relative files permit random access to their records. Each record is uniquely identified by its position in the file, RELATIVE to the start of the file. (See Figure 3.)



Each record is retrieved randomly by location RELATIVE to the first record in the file.
Example: Read record number 4.

Figure 3. Relative File

Let's return to our payroll example. If you desired to process the fourth record in the file (where you have programmatically kept track of the record number), you can retrieve the fourth record without processing records 1, 2, and 3 first.

By their very nature, relative files can only exist on random access devices such as DISCS.

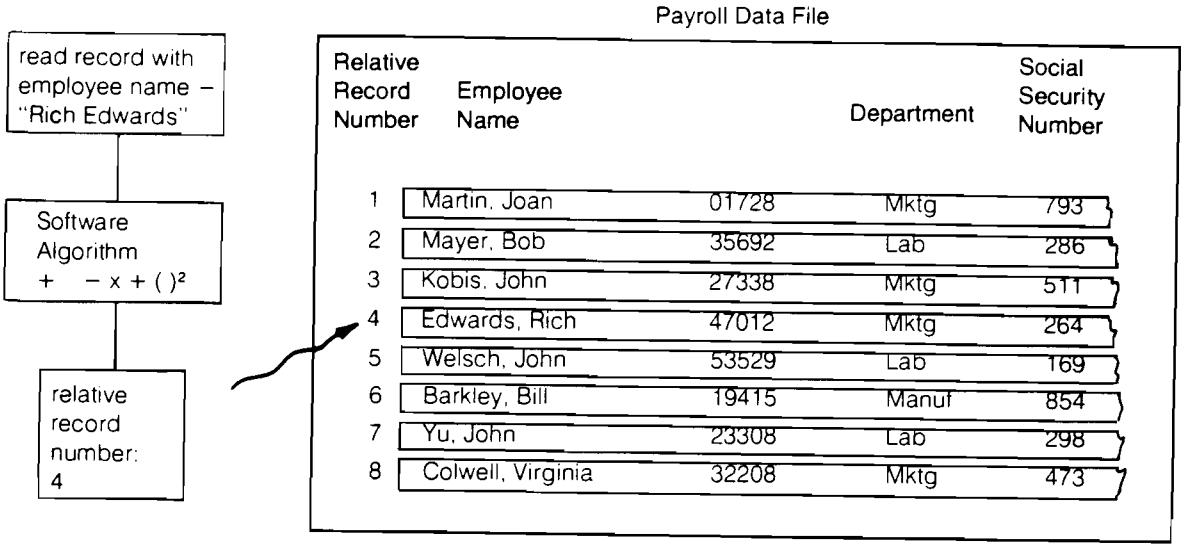
Note that it is possible to process the random file sequentially as we did with the sequential file – merely read the records in serial order, beginning with the first one.

Direct (Hashed) Files

Let's suppose that you want to be able to process a particular record based on the information stored within that record in the file. In our payroll example, you might wish to give a pay bonus to a particular employee, named Rich Edwards, but you didn't want to process every pay record (in a sequential file) and didn't know the record number (in a relative file). You'd like to be able to say: "Find me the record where the employee name = 'Edwards, Rich.'". Direct files allow you to process your data files in this manner.

Direct files permit retrieval of a record based upon the content of a data field within the record. In our payroll example, we chose the employee name to be the key. We might also wish to to locate a particular record based on the employee's social security number or perhaps his employee number.

A direct file is a file in which the records are accessed by relative record numbers which are calculated by a hashing algorithm. This method avoids the necessity of examining other records or knowing the relative record number in order to retrieve the desired data. While many different types of algorithms exist to convert an employee name (e.g., Edwards, Rich) into a relative record number (e.g.,record 4), we won't examine them here. For the purpose of understanding direct files it is enough to know that such algorithms exist and work. Figure 4 illustrates this type of file organization.



Each record is accessed by relative record number calculated by a software algorithm.

Example: Read the record in which employee name = "Edwards, Rich"

Figure 4. Direct File

The real advantage of direct files is that they allow the user to find records for processing based on information he is familiar with – employee name, social security number, employee number, for example – rather than having to keep track of the record's location within the file (relative record number) or process each record serially to locate the record of interest. **File maintenance is easy;** the key sequence is independent of the physical sequence. BECAUSE OF THEIR EXTREME FLEXIBILITY AND "FRIENDLY" USER INTERFACE, DIRECT FILES HAVE BECOME VERY POPULAR IN COMMERCIAL DATA PROCESSING. However, as you'll learn later here, there is a better approach.

File Access Methods

Records within a file can be processed one after another (sequentially), or without regard to the previous record (randomly). Processing on records includes such operations as reading, writing, updating and deleting. The types of access permitted to a file are determined by the organization of the file, as discussed in the preceding sections.

Sequential Access

In sequential access mode, records are processed in consecutive order. Sequential access is most appropriate for those files in which most if not all of the records are to be processed each time the file is opened. In a sequentially organized file, this is the only access method feasible. Records in a sequential file are read in the order in which they were written, starting with the beginning of the file. Records written to a sequential file follow (logically and physically) the previously written record.

Random Access

Random access is the access of a record in a file without processing preceeding records; i.e., without regard to the last record written or read.

To use this access method each record in the file must be identified by a unique relative record number. A record is then accessed by specifying this unique relative record number. In our payroll example, the user might request record number 4, for example. (See Figure 5 below.) An extension of this method is direct access which calculates the relative record number using a hashing algorithm.

	Employee Name	Employee Number	Department	Social Security Number
1	Martin, Joan	01728	Mktg	793
2	Mayer, Bob	35692	Lab	286
3	Kobis, John	27338	Mktg	511
→ 4	Edwards, Rich	47012	Mktg	264
5	Welsch, John	53529	Lab	169
6	Barkley, Bill	19415	Manuf	854
7	Yu, John	23308	Lab	298
8	Colwell, Virginia	32208	Mktg	473

Example: Read record number 4.

Figure 5. Random Access by Relative Record Number

Data Base Management Systems

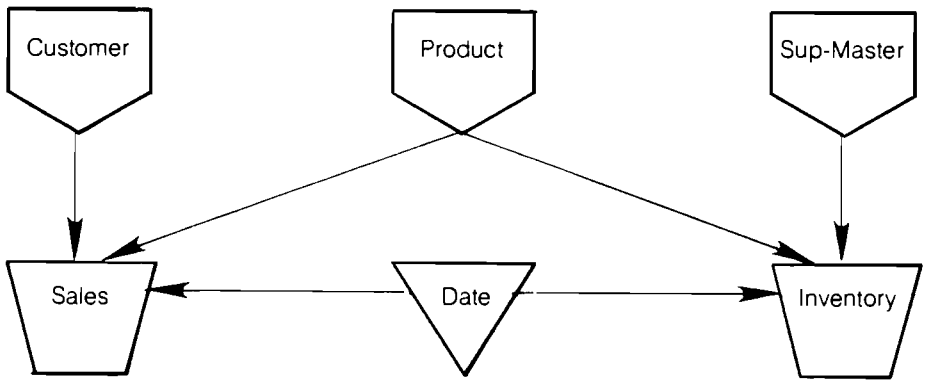
What Exactly is a Data Base Management System?

Let's first **define a DATA BASE as a collection of logically-related files containing both data and structural information. Pointers within the data base allow you to gain access to related data and to index data across files.** A key point to note is that a data base consists of MULTIPLE related files, whereas our preceding discussions on file organization and file access methods have been concerned with INDIVIDUAL files. (See Figure 6 and 7 on the following pages.)

A DATA BASE MANAGEMENT SYSTEM (DBMS) is a software package designed to operate interactively upon a collection of computer-stored files. It allows a user to define a data base structure and manipulate the contents by storing, retrieving, deleting, modifying and sorting data.

A typical DBMS includes a:

- Schema – special language that is used to decribe the location, contents, relationships, and security level of data that is stored in a data base.
- Data Manipulation Statements – are used to manipulate and extract data from the data base.
- QUERY – a special inquiry system incorporated into some DBMS that allows easy access to the data base for impromptu reports or data modification.
- Utility Programs – routines that are used to load the data base onto disc and to duplicate the data base as a backup.



Example: Related data sets (files)

Figure 6. Data Base

How Does a DBMS Differ from a File System?

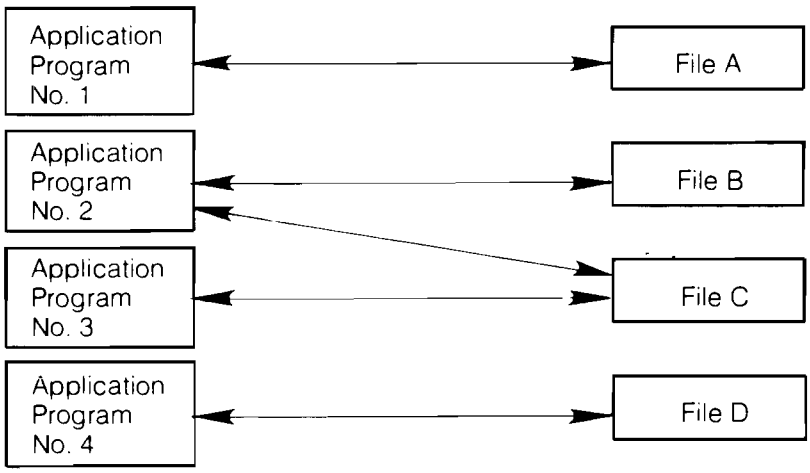
An examination of Figure 7 reveals several differences between a file system and a data base management system:

- In a file system individual files are not logically related; in a data base management system (DBMS) they are. BENEFIT: in a DBMS redundant data in related data sets (files) can be reduced.
- In a file system individual application programs must have a definition of the record format of the files; in a data base management system (DBMS) the system does this bookkeeping, transparent to the application programs. BENEFIT: in a DBMS, data is relatively independent of individual application programs.
- A file system requires an application program to retrieve data from the file unless a file copy utility is used; a DBMS typically includes an English-like inquiry facility for on-line data retrieval and report generation. BENEFIT: one-time or "what-if?" questions are easily answered without coding, debugging, and running an application program.

Other differences which aren't illustrated in Figure 7 include:

- Security passwords can exist at the data set level within a DBMS as compared with a more general security system in a file system. BENEFIT: the DBMS allows greater tailoring of access restrictions to implement data security.
- Schema presents a well-structured definition of the data base, including file interrelationships, record organizations, etc. BENEFIT: structure is no longer "hidden" within application programs; existing programs become easier to understand and modify.
- (Relates to above) Programs, to some extent, are independent of the data base. Changes in data base structure do not affect all programs – only those that access the altered portion. (The effect on these programs may be minimal.)

File System



Data Base Management System:

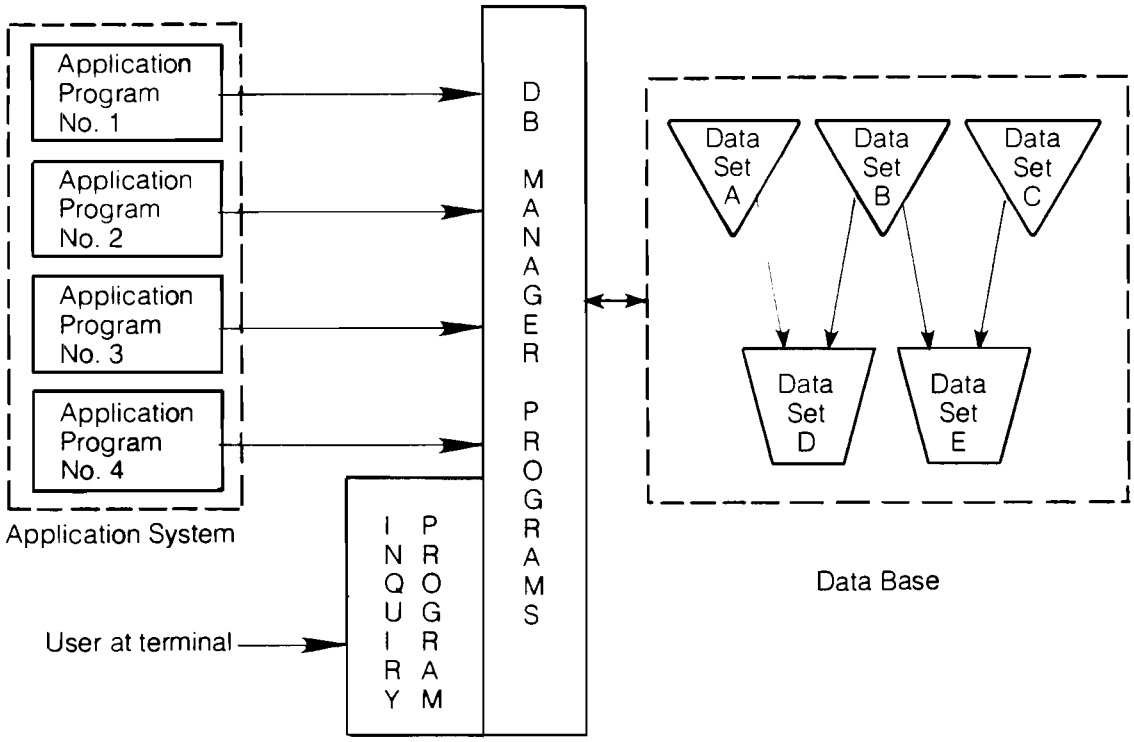


Figure 7. File System Compared to Data Base Management System

IMAGE/250 Overview

Every year the volume of information which must be maintained to efficiently manage a commercial, scientific, or educational enterprise grows by leaps and bounds. The development of computers as a management tool has been accompanied by a need for a more sophisticated means of information management and retrieval. Consequently, the concepts of computerized data bases and data base management have emerged.

As mentioned previously, DATA BASE MANAGEMENT implies both a standard, but flexible, data base structure, and a standard set of programs which are used for creating, accessing, and maintaining data bases. With the evolution of this concept, programmers can now focus their efforts on the design of the data base and on writing the application programs which use it, rather than on re-inventing the fundamental data base creation and the subsequent maintenance programs.

IMAGE, the HP 250 data base management system, provides the user with a powerful means of developing, accessing, and maintaining data bases tailored to today's corporate, industrial, scientific, and educational needs. QUERY, the HP 250 data base inquiry facility, provides the user with a simple means of examining or altering the content of a data base or of generating reports.

IMAGE provides the following data base management tools:

- A data base definition language with which to write a formal description of the data base.
- A schema processor, which translates this description into an internal description and stores it in a disc file.
- Procedures which allow the user to access the data from application programs.
- Special routines to sort the data base.
- Utility programs which maintain the data base by initializing, saving, restoring, or purging the data base.

IMAGE allows the data base designer to define a set of passwords which are used to prevent unauthorized read and write access to the various sets of the data base. It also has different access modes which determine the types of operations the user, and concurrent users, can perform on the data base.

IMAGE Data Base Structure

An IMAGE data base consists of data sets, data entries, and data items. A DATA ITEM is the smallest accessible data element. Each data item consists of a value referenced by a data item name. In general, many data item values are referenced by the same data item name. A data item corresponds to a "field" in a file system. An example would be an employee number in an employee record.

A DATA ENTRY is an ordered set of related data items. This corresponds to a record such as an employee record in a file system. The order of the data item values within a particular data entry is determined by the definition of that data entry as specified by the user when designing the data base. The length of a data entry is the combined lengths of all data items within it.

A DATA SET is a collection of data entries. A data set corresponds to a file, for example, an employee file. Each data set is referenced by a unique data set name. Figure 8 shows an example of three data entries in a data set named EMPLOYEE.

A DATA BASE is a named collection of related data sets. The data base is defined in terms of data items and data sets. A data base using IMAGE/250 may be defined with up to 255 data item names and 50 data set names. Figure 9 shows an example of a data base named COMPNY consisting of three data sets named LABOR, EMPLOYEE, and PROJECT.

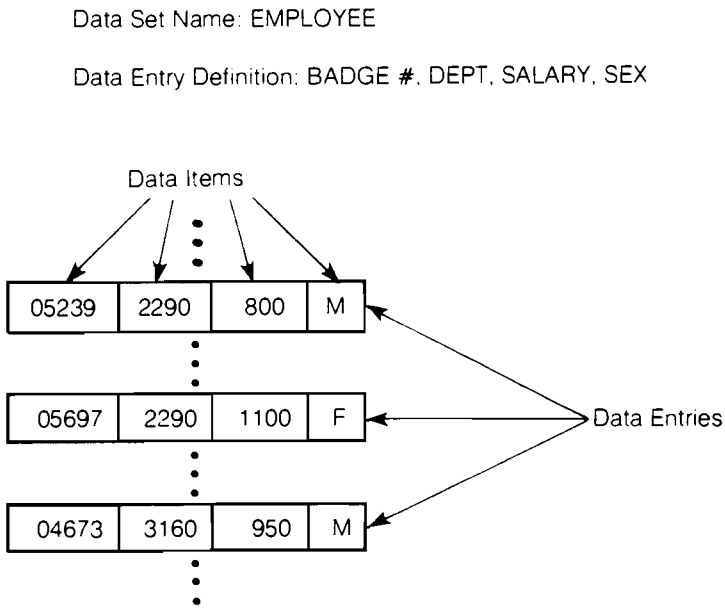


Figure 8. Sample Data Set

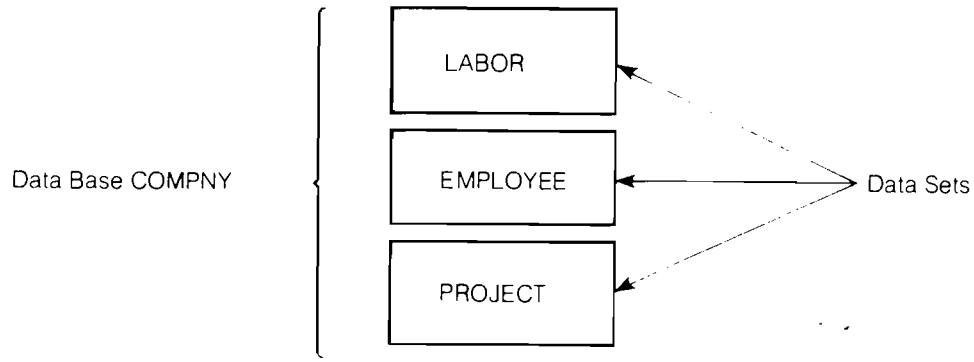


Figure 9. Sample Data Base

Types of Data Sets

An IMAGE data set is either a MASTER or a DETAIL data set.

Master data sets

- are used to keep information relating to a uniquely identifiable entity; for example, information relating to an employee (such as his name, identification number, address, and salary).
- allow for rapid retrieval of a data entry since one of the data items in the entry, called the SEARCH item, determines the location of the data entry.
- can be related to detail data sets containing similar search items and thus serve as indexes to the detail data set.

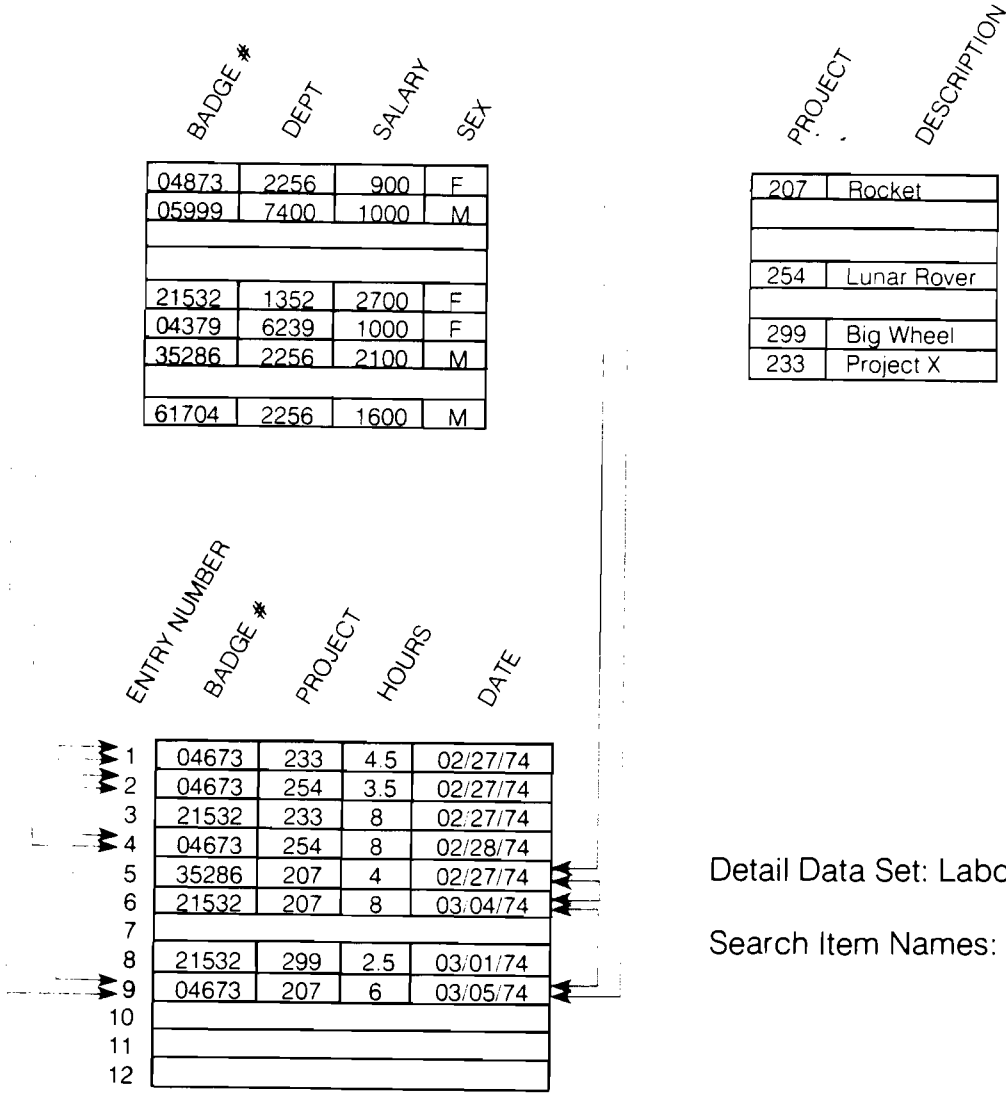
Detail data sets

- are used to record information about related events; for example, information relating to each pay period for each employee in the master data set.
- allow retrieval of all entries pertaining to a uniquely identifiable entity; for example, an employee identification number may be used to retrieve all data entries containing information about previous pay periods for an individual with that number.

Figure 10 illustrates the relationship of two master data sets to one detail data set. The master data set EMPLOYEE is related to the detail data set through the search item BADGE#. The master data set PROJECT-MASTER is related to the same detail data set through the search item PROJECT.

Master Data Set: Employee
Defines all values of Search item BADGE. Each entry relates to one or more detail set entries.

Master Data Set: Project-Master
Defines all values of Search item PROJECT.



Detail Data Set: Labor

Search Item Names: BADGE #, PROJECT

Figure 10. Master/Detail Data Set Relationships

IMAGE can rapidly retrieve all entries in the detail which have identical values for the search item. For example, the user can request all entries for the employee with BADGE #04673. The arrows indicate the relationship between a master data set entry and several detail data set entries.

Each master data set may serve as an index, singly or multiply, to one or more detail data sets. Each detail data set may be indexed by one or more master data sets (or none).

Using IMAGE

The overall use of IMAGE involves five general steps:

1. Creating an external data base description
2. Building the internal data base description
3. Building the data base files
4. Accessing the data base
5. Maintaining the data base

First the user describes the data base through the IMAGE Data Base Description Language. This description is called a SCHEMA. The schema describes all aspects of the data base structure including the data items, data sets, and data set relations. In addition, the schema describes the privacy and security provisions for the data base. Figure 11 illustrates a sample schema which a user would interactively create at his terminal console.

```
BEGIN DATA BASE COMPNY;
 PASSWORDS:
  3 TRICIA;
  7 WENDELL;
  9 NARTHEX;

 ITEMS:
  BADGE#,   I;
  DATE,     X8;
  DEPT,     X4;
  DESCRIP,  X16;
  HOURS,    I;
  PROJECT,  X4;
  SALARY,   L;
  SEX,      X2;

 SETS:
  NAME:     EMPLOYEE,MANUAL(3,9/7);
  ENTRY:    BADGE#(1),
            DEPT,
            SALARY,
            SEX;
  CAPACITY: 500;

  NAME:      PROJECT-MASTER,MANUAL(3,9/7);
  ENTRY:    PROJECT(1),
            DESCRIP;
  CAPACITY: 75;

  NAME:     LABOR,DETAIL(3,9/9,7);
  ENTRY:    BADGE#(EMPLOYEE),
            PROJECT(PROJECT-MASTER),
            HOURS,
            DATE;
  CAPACITY: 10000;
```

Password Definitions

Data Item Definitions

Data Entry and Data Set Definitions

Figure 11. Sample of IMAGE Data Base Description Language

To build an internal description of the data base, the user invokes the IMAGE Schema Processor. The Schema Processor accepts the schema as input and creates an internal representation of the schema in a disc file known as the ROOT FILE.

The user then invokes an IMAGE utility program to build empty data files on disc, one for each defined data set.

Finally the user's application programs call IMAGE statements to enter, retrieve, and modify data base content.

Once the data base application is operational, the user creates back-up copies and performs other maintenance tasks with the IMAGE utility programs.

Accessing the Data Base

The user accesses a data base through statements in his BASIC program. These statements locate data, maintain pointer information, manage the allocated file space, and return status information to the user. The tasks performed by the IMAGE statements relieve the user of the "bookkeeping" normally associated with file management and allow him to concentrate on processing his application. Figure 12 shows a sample statement as it might appear in an application program. Table 1 briefly summarizes the available DBMS statements.

Some of the functions accomplished through the use of IMAGE statements are:

- Adding a new entry to a data set.
- Deleting an entry from a data set.
- Reading a data entry.
- Changing the values of data items of an entry.
- Selecting and sorting certain entries.

When using an IMAGE statement, the user must pass information about the particular data requested in the form of a parameter list; for example, the data base and data set names.

```
DBGET (base name, data set, mode, status, list, buffer, argument)
```

Figure 12. Sample Statement

The user need not be aware of the physical location of the entry itself.

For example, the DBGET procedure retrieves an entry as it is stored in the data base and transfers it to the user's buffer area.

Table 1. DBMS Statements

Statement	Function
DBOPEN	Initiates access to a data base.
DBINFO	Provides information about the data base currently being accessed (such as the name and description of a particular data item).
DBCLOSE	Terminates access to a data base.
DBFIND	Locates the first entry of a data chain in preparation for access to other data entries in the chain.
DBGET	Reads the data items of a specified entry.
DBUPDATE	Modifies data set entries by changing the value of data items which are not search items.
DBPUT	Adds new entries to a data set.
DBDELETE	Deletes existing entries from a data set.
DBLOCK	Locks a data base temporarily to allow exclusive access.
DBUNLOCK	Unlocks a data base which was locked by a previous DBLOCK.

At any given time, a record may or may not contain an entry. IMAGE maintains internal information indicating which records of a data set contain entries and which do not.

Each time a request for retrieval is made, the IMAGE statement must determine the data entry from which to extract the desired data items. The determination is based upon the type of access requested in the IMAGE statement. There are four modes of access available:

- SERIAL – In this mode, IMAGE starts at the most recently accessed record and sequentially examines successive records until the closest data entry, if any, is found.
- DIRECTED – In this mode, the program specifies the record address of the data entry to be retrieved.
- CALCULATED – This type of access is retrieval of master data set entries based upon the search item (key) value. It is not necessary for the user to know where the desired entry is located within the data set. IMAGE uses the search item value as input to an address formula to calculate the record number of the desired entry.
- CHAINED – This type of access consists of successive retrieval of all entries in a detail data set which contain the same search item value.

The example back in Figure 10 can be used to illustrate chained access. All pieces of work done by the employee whose badge number is 04673, or all the work done on project 207, can be rapidly retrieved using chained access. By contrast, a program using a non-search item such as DATE to locate entries containing specific DATE values must serially search the data set. The selection of data items as search items depends upon expected usage, frequency of retrieval, number of distinct values, and other criteria.

Maintaining the Data Base

The user initializes and maintains a data base by executing IMAGE Data Base Utilities. These programs help maintain data base integrity.

IMAGE utilities also perform the following functions:

- Copy the data base from disc files to back-up discs, thus providing back-up copies of the data base.
- Reload a data base from back-up discs back into the data base disc files.
- Erase data base content while maintaining the root file and the data files of the data base.
- Purge the entire data base including its root file and data files.

The user can copy either the data base content or the data base content and related structural information to a back-up disc. The first option allows the user to store the content of a data base on a back-up disc and then restore the data into a new data base structure. To do this, the user first stores the content of a data base on a back-up disc, using the DBUNLD program, then redefines the data base structure (using the definition language and the Schema Processor), and finally restores the data from a back-up disc into the new data base structure using the DBLOAD program.

Table 2. Utilities

Utility	Function
DBCREATE	Allocates and initializes all data sets of a data base.
DBERASE	Reinitializes all data sets of a data base.
DBPURGE	Purges the root file and all data sets of a data base.
DBSTORE	Stores the root file and all data sets of a data base on a back-up disc.
DBRESTORE	Restores the root file and all data sets from a back-up disc created by the DBSTORE program to an on-line disc.
DBUNLD	Dumps all the data entries of each data set of a data base to a specially formatted back-up disc. This logical dump of the data base is useful for changing the data base structure.
DBLOAD	Loads an already created data base with the content of a DBUNLD back-up disc.

QUERY ... The Data Base Inquiry Facility

QUERY provides a simple method of accessing an IMAGE data base without programming effort. QUERY allows users to do the following:

- add data entries
- modify or delete data entries
- retrieve data which meets selection criteria
- report on the data retrieved

To perform these operations, the user enters simple commands consisting of English-language key words such as FIND and LIST.

The user only needs to know the relationships of the data base elements and not the structure of the disc files; QUERY finds the data and performs the operations in response to commands using the data set and data item names specified.

QUERY is used from the console and prompts the user for commands, issues error messages when an error occurs, and prints other information of interest to the user.

QUERY output can be directed to the console or a line printer.

Applications

QUERY can be used for a variety of applications, complex or simple. Some of these applications are:

- **Casual inquiry of the data base** – because QUERY requires no programming effort, the user can search a data base for information without writing a program or waiting for an applications/report program to be run. The user defines "English-like" commands representing what he wants.
- **On-line data updates** – with QUERY the user can enter, delete and modify data on-line, directly from the console to the IMAGE data files. No intermediate steps need be taken. The data base is updated quickly; the user need not wait for periodic update runs to enter the latest information into his data base.
- **Report generation** – QUERY's automatic report formatting produces reports with header and column labels, page numbers, and group labels. In addition, the user can sort entries through multiple fields, as well as total columns of numeric data values.

  Application program debugging – if the user is developing programs which access IMAGE data bases, QUERY makes an excellent debugging aid. The user can manipulate data in the data base using his program and then use QUERY to access data in the data base to test the results.

Features

In addition to the features mentioned above, QUERY offers the following:

- **Selection of data through logical comparisons** – QUERY locates entries based upon logical criteria specified in the FIND command. Many conditions can be specified in one command. For example, the command: FIND COMMISSION>"2000" and OCCUPATION = "SALESMAN" instructs QUERY to locate those data entries with the value of data item COMMISSION greater than 2000 and OCCUPATION equal to SALESMAN. Only those entries which fulfill the two criteria are located.
- **Obtain information from several sets** – several data sets can be accessed together using a command called THREAD.
- **Display of the data base structure** – the data base structure can be displayed through the use of the INFO command. QUERY prints out the data items and data sets you may access and shows the relationship between them. Only those sets to which you have at least read access are displayed.
- **Storing commands in a file** – a sequence of commands that you wish to rerun can be stored in a file and executed at any time.
- **Security provisions** – QUERY adheres to all of the security provisions included in the IMAGE data base by the data base designer. After QUERY is invoked, the user must enter a designer-supplied password. This word determines which data sets the user may access in the data base. Additional security is available in QUERY for those data items which the user decides should not be modified in QUERY.

Command Set

The QUERY command set consists of numerous commands. Table 3 shows some of them and summarizes what they do.

Table 3. QUERY Command Set

Category	Commands	Function
Environment	DATA BASE	Specifies the data base to be accessed.
	OUTPUT TO	Specifies the ouput device to be used.
	PASSWORD	Specifies access password.
Utility	INFO	Displays the structure of the data base being accessed.
	EXIT	Terminates QUERY execution.
Location	THREAD	Ties several data sets together.
	FIND	Locates data entries in the data base according to your specifications.
Sorting	SORT	Sorts data entries located by FIND.
Updating	ADD	Adds data entries to the data base.
	DELETE	Removes data entries from the data base.
	REPLACE	Modifies data items in data entries.
Reporting	LIST	Prints entries with automatic formatting.
Operation	DO	Executes QUERY commands from a file.

CONCLUSION

The previous section should have given you a good understanding of what DATA BASE MANAGEMENT is and the power it has to offer. You will have a chance to learn more about and actually use a data base in the coming course.

HEWLETT PACKARD

SALES & SERVICE OFFICES

UNITED STATES

ALABAMA
8290 Whitesburg Dr S E
P O Box 4207
Huntsville 35802
Tel (205) 881 4591

Medical Only
228 W Valley Ave
Room 220
Birmingham 35209
Tel (205) 942 2081 2

ARIZONA
2336 E Magnolia St
Phoenix 85034
Tel (602) 244-1361

2424 East Aragon Rd
Tucson 85706
Tel (602) 294-3148

***ARKANSAS**
Medical Service Only
P O Box 5646
Brady Station
Little Rock 72215
Tel (501) 376-1844

CALIFORNIA
1430 East Orangethorpe Ave
Fullerton 92631
Tel (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel (213) 877-1282
TWX 910-499-2671

5400 West Rosecrans Blvd
P O Box 92105
World Way Postal Center
Los Angeles 90009
Tel (213) 970-7500

***Los Angeles**
Tel (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel (408) 249-7000
TWX 910-338-0518

***Ridgecrest**
Tel (714) 446-6165

646 W North Market Blvd
Sacramento 95834
Tel (916) 929-7222

9606 Aero Drive
P O Box 23333
San Diego 92123
Tel (714) 279-3200

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel (303) 771 3455

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel (203) 389-6551
TWX 710-465-2029

FLORIDA
P O Box 24210
2806 W Oakland Park Blvd
Ft. Lauderdale 33311
Tel (305) 731-2020

***Jacksonville**
Medical Service only
Tel (904) 398-0663

P O Box 13910
6177 Lake Ellenor Dr
Orlando 32809
Tel (305) 859-2900

P O Box 12826
Pensacola 32575
Tel (904) 476-8422

GEORGIA
P O Box 105005
Atlanta 30348
Tel (404) 955-1500
TWX 810-766-4890

Medical Service Only
***Augusta** 30903
Tel (404) 736-0592

P O Box 2103
Warner Robins 31098
Tel (912) 922-0449

HAWAII
2875 So King Street
Honolulu 96814
Tel (808) 955-4455
Telex 723-705

ILLINOIS
5201 Tollview Dr
Rolling meadows 60008
Tel (312) 255 9800
TWX 910-687 2260

INDIANA
7301 North Shadeland Ave
Indianapolis46250
Tel (317)842-1000
TWX 810-260-1797

IOWA
2415 Heinz Road
Iowa City 52240
Tel (319) 338-9466

KENTUCKY
Medical Only
Atkinson Square
3901 Atkinson Dr
Suite 407 Atkinson Square
Louisville 40218
Tel (502) 456-1573

LOUISIANA
P O Box 840
3229-39 Williams Boulevard
Kenner 70063
Tel (504) 443-6201

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel (301) 944 5400
TWX 710-862-9157

2 Choke Cherry Road
Rockville 20850
Tel 301) 948-6370
TWX 710-828-9684

MASSACHUSETTS
32 Hartwell Ave
Lexington 02173
Tel 617) 861 8960
TWX 710-326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel (313) 476-6400

*24 West Centre Ave
Kalamazoo 49002
Tel (606) 323 8362

MINNESOTA
2400 N Prior Ave
St. Paul 55113
Tel (612) 636-0700

MISSISSIPPI
***Jackson**
Medical Service only
Tel (601) 982-9363

MISSOURI
11131 Colorado Ave
Kansas City 64137
Tel (816) 763-8000
TWX 910-771-2087

1024 Executive Parkway
St. Louis 63141
Tel (314) 878-0200

NEBRASKA
Medical Only
7171 Mercy Road
Suite 110
Omaha 68106
Tel (402) 392-0948

NEW JERSEY
W 120 Century Rd
Paramus 07652
Tel (201) 265-5000
TWX 710-990-4951

Crystal Brook Professional Building
Eatontown 07724
Tel (201) 542-1384

NEW MEXICO
P O Box 11634
Station E
11300 Lomas Blvd N E
Albuquerque 87123
Tel (505) 292-1330
TWX 910-989-1185

156 Wyatt Drive
Las Cruces 88001
Tel (505) 526-2484
TWX 910-9983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel (518) 458-1550

201 South Avenue
Poughkeepsie 12601
Tel (914) 454-7330
TWX 510-253-5981

65C Perinton Hill Office Park
Fairport 14450
Tel (716) 223-9950

5858 East Molloy Road
Syracuse 13211
Tel (315) 454-2486
TWX 710-541-0482

1 Crossways Park West
Woodbury 11797
Tel (516) 921-0300
TWX 710-990-4951

NORTH CAROLINA
P O Box 5188
1923 North Main Street
High Point 27262
Tel (919) 885-8101

OHIO
16500 Sprague Road
Cleveland 44130
Tel (216) 243-7300
TWX 810 423-9430

330 Progress Rd
Dayton 45449
Tel 513 859-8202

1041 Kingsmill Parkway
Columbus 43229
Tel (614) 436-1041

OKLAHOMA
P O Box 32008
Oklahoma City 73132
Tel (405) 721-0200

OREGON
17890 SW Lower Boones Ferry Road
Tualatin 97062
Tel (503) 620-3350

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel (412) 782 0400

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel (215) 265-7000
TWX 510-660-2670

SOUTH CAROLINA
6941-O N Trenholm Road
Columbia 29260
Tel (803) 782-6493

TENNESSEE
***Knoxville**
Medical Service only
Tel (615) 523-5022

3027 Vanguard Dr
Director's Plaza
Memphis 38131
Tel (901) 346-8370

Nashville
Medical Service only
Tel 615) 244-5448

TEXAS
P O Box 1270
201 E Arapaho Rd
Richardson 75080
Tel (214) 231-6101

10535 Harwin Dr
Houston 77036
Tel (713) 776-6400

205 Billy Mitchell Road
San Antonio 78226
Tel (512) 434-8241

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel (801) 972 4711

VIRGINIA
P O Box 12778
No 7 Koger Exec Center
Suite 212
Norfolk 23502
Tel (804) 461 4025 6

P O Box 9669
2914 Hungary Springs Road
Richmond 23228
Tel (804) 285-3431

WASHINGTON
Bellefield Office Pk
1203 114th Ave S E
Bellevue 98004
Tel (206) 454-3971
TWX 910-443-2446

***WEST VIRGINIA**
Medical Analytical Only
Charleston
Tel (304) 345-1640

WISCONSIN
9004 West Lincoln Ave
West Allis 53227
Tel (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office nearest you Atlanta Georgia North Hollywood California Rockville Maryland Rolling Meadows Illinois Their complete addresses are listed above

***Service Only**

1/78

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd
11620A - 168th Street
EdmontonT5M 3T9
Tel (403) 452-3670
TWX 610-831-2431

Hewlett-Packard (Canada) Ltd
210 7220 Fisher St S E
Calgary T2H 2H8
Tel (403) 253-2713
Twx 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd
837 E Cordova Street
Vancouver V6A 3R2
Tel (604) 254-0531
TWX 610-922-5059

MANITOBA
Hewlett-Packard (Canada) Ltd
513 Century St
St James
Winnipeg R3H 0L8
Tel (204) 786-7581
TWX 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd
800 Windmill Road
Dartmouth B3B 1L1
Tel (902) 469-7820
TWX 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd
1020 Morrison Dr
Ottawa K2H 8K7
Tel (613) 820-6483
TWX 610-563-1636

Hewlett-Packard (Canada) Ltd
6877 Goreway Drive
Mississauga L4V 1M8
Tel (416) 678-9430
TWX 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd
275 Hymus Blvd
Pointe Claire H9R 1G7
Tel (514) 697-4232
TWX 610-422-3022
TLX 05-821521 HPCL

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada) Ltd in Mississauga

Pre-Study Workbook

HEWLETT PACKARD

MANUAL PART NO. 45104-30025

PRINTED IN U.S.A.
APRIL 1979