9.0 COMMANDS

The explicit definitions of the dBASE commands are in this section. The user should familiarize him/herself with these fundamentals before reading the rest of the command information.

9.1 SYMBOL DEFINITIONS

Understanding what the special symbols in the general formats of the dBASE commands really mean is vitally important. Not only does it help in understanding just what the form of the command really, it helps to show the potential of each command. Please read the following table throughly.

Symbol	Meaning
<commands> or <statements>	- means any valid dBASE statements; it also means whole statements. An IF without an ENDIF, (or a DO WHILE without an ENDDO), is only half of a statement, while a REPORT is a whole statement in itself.
<char string> or <cstring>	- means any character string; character strings are those characters that are enclosed in single quotes ('), double quotes ("), or square brackets ([ ]).
<delimiter>	- means any special character; special characters are those characters from the keyboard that are punctuation marks, like any one of the following "()#=,&.
<exp>	- means an expression; an expression can be created by tacking together numbers, functions, field names or character strings in any meaningful manner. "4+8", and "doc = '3' .or. doc = '4'", are both expressions as well as "$('abc'+&somestr,n,3) = 'abcdefg'".
<exp list>	- means a list of expressions separated by commas; usually simple expressions are used. Two of the examples in the previous paragraph are rather complicated, the first one could be considered as simple.
<field>	- means any record field name; in one of the examples that are in the following commands, one of the databases has field names like ITEM, COST, DATE, etc.
<field list> or <list>	- means a list of record field names separated by commas.

<file> or - means any filename; these are file names that
<file name>must obey the rules for file names that were stated in section 3.0.

<form file> - means the name of a report form filename; see section 3.4 and the REPORT command for the how and why of this type of file.

<index file> - means the name of the file where indexing information is placed; see section 3.6 and the INDEX command for the how and why of this type of file.

<key> - means the field name which will be indexed on; keys are important. There may be several indexes for any given database, each on different (or on a combination of) keys. Keys may be <expressions> or field names. See the INDEX command for more information.

<memvar> -- means any memory variable; memory variables are those variables that are created by STOREs or by use of a command that saves some value for later use (ACCEPT, INPUT, etc.) There is a maximum of 64 memory variables allowed in dBASE.

<memvar list> - means a list of memory variables separated by commas.

<n> - means a literal; literals are numbers which are not gotten from memory variables or calculations."4+8" is not a literal, while "4" and "9876" are literals.

<scope> - means a specification of the scope of the command; scope means how much does the command cover. There are three values that <scope> may take on.

ALL - means all the records in the file. All means that the file is rewound and whatever the command ALL the records in the file are searched for compliance. ALL is the default for some of the commands. For other commands the default will be the current record (specially for the more potentially destructive commands like DELETE). Each command description tell what is the default scope. In the case of using a FOR phrase in any of the commands, ALL will be the default.

NEXT n - means the next n records, including the current record; NEXT also begins with the record currently being pointed at. And n must have a literal value, that is, it must not be a memory variable or an expression.

RECORD n - means only record n; again, n must not be a memory variable or an expression--it must be literal before it will work.

FOR <exp> - Any record so long as some logical expression has a true value. Unless otherwise specified, the presence of a FOR clause causes ALL records to scanned (with a rewind of the database).

WHILE <exp> - All sequential records as long as some logical expression (<exp>) has a true value. The controlling command stops the first time the expression is false. The presence of a WHILE clause implies NEXT 65534 unless otherwise specified and does not rewind the database.

There are other special symbols used in the command formats. These are special to the command and will be explained in the body of the command.

9.2 RULES TO OPERATE BY

As with all command "languages" there are a set of rules which must be followed to successfully operate the program. The following rules are to use in translating the general format of the commands into the more useful specific forms.

1. The verb of any command must be the first non-blank character of the command line; the phrases may follow in any order. A verb is an action word; CREATE, APPEND, REPORT, SET, DISPLAY, and ERASE are all examples of verbs--they cause a specific action. Phrases are equivalent to adverbs; they more fully describe the action. FOR, NEXT, and WITH are examples of words that begin phrases. All of these example words are refered to as "keywords".

2. Any number of blanks may be used to separate words and phrases. Remember though, blanks are counted in the 254 limit described in Rule #3.

3. All commands must be less than 254 characters in length (even after a macro expansion).

4. Commands and keywords can be abbreviated to the first four (or more) characters. E.g. DISPLAY STRUCTURE could be input as DISP STRU or DISPL STRUCT or etc. Just remember that the abbreviation must also be spelled correctly up to the point where it ends.

5. Either upper or lower case letters may be used to enter commands, keywords, field names, memory variable names, or file names.

6. Parts of the commands are optional, that is, some parts of the commands may be left off when the command is used. Square brackets ([]) are used in the command formats to show which phrases are the optional constructs that may be left off. These are the phrases which are used to modify the action of commands. The upper case words are the keywords and they must be entered whenever the phrase that contains them is used.

7. A reserved word is a keyword that will generate an error if is

7. A reserved word is a keyword that will generate an error if is used for something other that what it is supposed to be. There are no reserved words in dBASE. However, certain field names and file names can cause difficulty, e.g., a command file named WHILE will be incorrectly interpreted as a DO WHILE statement by the DO command processor, ALL as a field name cannot be used in a number of commands. In general, it is a good practice to avoid the use of dBASE keywords as field names or file names.

8. dBASE statements in a command file must nest correctly. To nest something means that one statement must fit inside another statement. This is especially important to proper execution of the IF-ELSE-ENDIF and the DO WHILE-ENDDO groups. Indenting a command file will show if the statements are correctly nested. dBASE does not catch nesting errors, it will however execute the command file in an unknown manner. Below are examples of how to correctly nest these two statements.

```
DO WHILE .NOT. EOF

  statements
  .
  IF A .AND. B
    .
    more statements
    .
  ELSE
    .
    DO WHILE A <= 57
      .
      some more statements
      .
    ENDDO
    .
    even more statements
    .
  ENDIF
  .
  infinitely more statements
  .
ENDDO
.
```

This is the correct way to nest. The IF-ELSE-ENDIF statement is totally within the DO WHILE-ENDDO statement. Just as the second DO WHILE-ENDDO statement is totally within the ELSE part of the IF-ELSE-ENDIF. It would be just as easy to show more levels of nesting, since dBASE allows many more levels to exist.

```
DO WHILE .NOT. EOF
  .
  statements
  .
  IF something changes values
  .
  ENDDO
  .
  more statements
  .
  ENDIF
```

This is an example of a NO NO. The ENDDO crossed over the boundary of the IF-ENDIF group, that is, the two statements do not nest properly. The command file that holds these statements will not work as expected AND dBASE will not explain why.

?
-

```
? [<exp list>]
?? [<exp list>]
```

This command is a specialized form of the DISPLAY command; it is equivalent to DISPLAY OFF <exp>. It can be used to show the value an expression or list of expressions. The question mark command (possibly pronounced "what is" can use memory variables, database fields, constants, or functions. A "?" with no expression spaces down a line on the output. This feature is particularly useful in command files to "open up" the displays.

The second form of this command "??" behaves like a single "?" except that no line feed or carriage return is done before the expression is printed. This can be used in command files to output more than one expression to the same output line.

Examples:

```
. USE EXAMPLE
. 4
. ? #
 4
. ? NAME
CHANG, LEE
. ? 5+9
 14
```

Following is a sample command file that uses the ? to space out the display. The command file is set up to be executed with the command: "DBASE H:FILE". The dBASE response to the command file follows the command file.

```
set default to g
use trace index trace
disp stru
?
accept "Enter today's date." to dte
set date to &dte
release dte
return
```

```
STRUCTURE FOR FILE:   TRACE.DBF
NUMBER OF RECORDS:    02359
DATE OF LAST UPDATE:  10/08/81
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001       UP           C     024
002       TRFLD        C     005
003       DOC          C     024
004       DESCR        C     080
005       NATURE       C     010
006       STATUS       C     006
007       TESTED       C     004
** TOTAL **                00154

Enter today's date.:10 14 81
```

@

```
@ <coordinates> [SAY <exp> [USING <format>]]
        [GET <variable> [PICTURE <format>]]
```

This command works with the SET FORMAT TO, ERASE, EJECT, CLEAR GETS and READ commands and is a most powerful way to display specific, formatted information on the screen or the printer. The way an "@" is interpreted changes according to how the SET FORMAT TO command is used. Also whether or not one of the other commands has an effect also depends on the SET command. All combinations are discussed below.

The <coordinates> are an "x,y" pair and may take on one of two meanings, either they are screen coordinates or they are printer coordinates. The "x,y" denotes line (x) and column (y). On most CRTs, the screen oriented coordinates have an "x" range of 0-23, and a "y" range of 0-79, that is 24 lines by 80 columns. dBASE uses the 0th line for messages to the user and the user should avoid using it. The printer oriented coordinates have both an "x" and a "y" range of 0-254. For either of these two meanings the coordinates can be any literal, numeric memory variable, or numeric expression. The SET FORMAT command is used to choose between either of these two meanings.

When a SET FORMAT TO SCREEN command has been issued (which is the default), the "@" command causes data to be displayed on the screen. A coordinate pair of 0,0 means the first character location on the upper left corner of the display. (This frequently referred to as the home position.) The pair 10,15 means the 11th line and the 16th column of the display. Again the 0th line on the screen should not be used. "@" commands may be issued in any order to the screen. That is, one may SAY something to line 15 before one SAYs something to line 10. Likewise columns may be filled in any order.

When a SET FORMAT TO PRINT command had been issued, the "@" command will cause data to be printed on the printer. The coordinate pair 0,0 refers to the upper left hand corner of the paper. "@" commands to the printer must be output in order. Much paper will be wasted if this is not done. The user may like to pretend that a typewriter is being use (indeed, it is). All commands to line 5 must preceed commands to line 6, also, all commands to column 10 must preceed commands to column 20, etc. If this is not done a page eject will occur before the new line is printed.

When the SET FORMAT TO SCREEN has been issued, an ERASE will clear the screen of all information that was previously on it, will release all the GETs (see below), and will reset the coordinates to 0,0. When the SET FORMAT TO PRINT has been issued an EJECT will do a page feed and reset the coordinates to 0,0.

The SAY phrase is used to display an expression that will not be altered by subsequent editing via the READ command. The USING subphrase is used to format the expression emitted by the SAY phrase. Formatting directives are explained below. It is a good thing to always use the USING subphrase. dBASE will take liberties with the expression if there is no USING.

SAY phrases may be used on either the screen or the printer. GETs however, will only be recognized when the SET FORMAT TO SCREEN command has been issued.

The GET phrase displays the current value of a field variable or memory variable. The variable must exist prior to issuing of the GET and is subject to later editing by the READ command. The PICTURE phrase may be used with a GET phrase to allow special formatting and validation of the data as it is entered (see the READ command for further information). If no PICTURE clause is given, then the data type (character, numeric or logical) forms an implicit PICTURE.

If the data type of the field variable or memory variable in the GET is logical then the data validation allows only the characters 'T', 'F', 'Y', 'N' and their lower case equivalents to be entered.

A maximum of 64 GETs can be active at any given time. Either the ERASE command or the CLEAR GETS command may be used to release the existing GETs.

When SET FORMAT TO SCREEN is in effect and if neither a SAY or a GET phrase is given, then the remainder of the line indicated by the coordinates is cleared to spaces. Thus @ 10,0 will clear the entire 11th line.

When the SET FORMAT TO SCREEN is in effect, a READ must be issued in order to "fill" the GETs. (See the READ command). However when SET FORMAT TO PRINT is in effect, "@" commands require no subsequent READ commands to complete their action.

Not needing a READ to print allows the user to directly format the output for any pre-printed material (such as checks, purchase orders, etc.) in a most convenient manner. The user need only to remember that "@" commands must be issued as if one were typing on a typewriter.

In using the SET FORMAT TO PRINT capability, it is often necessary to print out more than one item. The ability to subsitute memory variables for the coordinate values is important. The following example is from a command file that generates a special report form for a special task.

```
SET FORMAT TO PRINT
GOTO TOP
STORE 7 TO CNTR
DO WHILE .NOT. EOF
   IF CNTR >= 50
      EJECT
      STORE 7 TO CNTR
   ENDIF
   @ CNTR,12 SAY P USING 'XXXXXXXXXXXXXXXXXX ʌʌ..  XXXXXX'
   @ CNTR,48 SAY D USING 'XXXXXXXXXX'
   @ CNTR,64 SAY P1 USING 'XXXXXXXXXXXXXXXXXX
   @ CNTR,88 SAY U USING 'XXXXXXXXXX'
   @ CNTR,104 SAY P2 USING 'XXXXXXXXXXXXXXXX)
   IF RCD <> 0
      @ CNTR,130 SAY RCD USING '9999'
   ENDIF
   STORE CNTR + 1 TO CNTR
   SKIP
ENDDO
RETURN
```

In this command file, a maximum of 57 lines will be printed on the printer before a page eject is done. The purpose here was to print out most of the fields of a database (and selectively print out one of the fields). Care must be taken to make sure enough room is given to the SAY phrase to emit the variable. If the USING is shorter than the variable or the field, the variable or field is truncated. The <format> for the USING (the 'XXX...X' strings are explained in the table below.

Also, in the SET FORMAT TO PRINT mode, if the coordinates of the next "@" allow information to be printed on the same line but start it in a column that has already been printed, the printer may not output the proper information. In fact, the printer may go to the extreme right and print (in one square) all the information in the rest of the line. In the SET FORMAT TO SCREEN mode, the old information will be written over by the new information.

The last form of the SET FORMAT command is: SET FORMAT TO <format file>. When this command is in effect and when a READ command has been issued, the "@" commands are READ from the pre-designed <format file>. In this manner the user may design the screen into a format for more specialized purposes. It is important to note here that the use of format files is not necessary for use of "@"s, since "@"s may reside in command files. See READ for more information.

Formats:

Both the USING and PICTURE clauses have as their object, a format. The format is a series of characters that indicate which characters appear on the screen or page. The following table defines the characters and their functions:

Format character	SAY function	GET function
#	causes the next number to be output	allows only a digit (1,2,...,8,9,0) and the characters ".", "+", "-", and " " (a space) to be entered
9	same as #	same as #
X	outputs the next character	allows any character to be entered
A	outputs the next character	allows only alpha. to be entered
$ or *	outputs either a digit or a $ or * instead of leading zeros	output as is
	no effect	converts lowercase alpha characters to uppercase

Example:

```
. @ 5,1 SAY 'ENTER PHONE NUMBER' GET PNO PICTURE '(999)999-9999'
```

The message 'ENTER PHONE NUMBER' would be displayed, followed by '(bbb)bbbb-bbbb' (b indicates a blank) assuming that the value of PNO was all blanks prior to issuance. When (and if) the READ command is issued, only digits can be entered. The value of PNO after the READ command might well be '(213)555-5555' after editing. All of the non-functional characters in the PICTURE format are inserted into the variable. In this example, the parentheses, minus sign and the blank are non-functional.

```
. @ 10,50 SAY HOURS*RATE USING '$$$$$$$.99'
```

This "@" command could be used with either the screen or the printer since it has no GET phrase. It might well be used to print payroll checks. The dollar signs will be printed as long as there are leading zeros in the item to be printed. If hours=40 and rate = 12.50 then '$$$$500.00' will be displayed. This feature is known as floating dollar and is valuable for printing checks that cannot be easily altered in value.

When commas are used in the integer part of a picture, they are replaced by the picture character in front of them if there are no significant digits in the item to the left of where the comma would otherwise be placed.

```
@ 10,50 SAY HOURS * RATE USING '$$$,$$$.99'
```

Would output $$$$500.00 and specifically not output $$$,500.00.

Normally, a number of "@" commands are issued then, if any GET phrases were included, a READ command is issued to allow editing or data entry into the GET variables. In the following example the screen is formatted with several "@"s and a database is filled with information according to these "@"s. The last record in the database will have a "0" in the field "name", this is the record that will be deleted, since it is not necessary.

```
SET FORMAT TO SCREEN
USE F:EXAMPLE
ERASE
DO WHILE NAME # '0'
  APPEND BLANK
  @ 5,0 SAY "ENTER NEXT NAME" ;
        GET NAME PICTURE 'XXXXXXXXXXXXXXXXXXXX'
  @ 6,0 SAY "ENTER TELEPHONE NUMBER";
        GET TELE:EXTSN PICTURE 'XXXXX'
  @ 6,40 SAY "ENTER MAIL STOP" ;
         GET MAIL:STOP PICTURE 'XXXXXXXXXX'
  READ
ENDDO
GOTO BOTTOM
DELETE
PACK
LIST
RETURN
```

The following commands affect the operation of the "@" command:

* SET INTENSITY ON/OFF (default is ON) affects the screen intensity of GET's and SAY's.

* SET BELL ON/OFF (default is ON) affects the bell alarm when invalid characters are entered or a data boundary is crossed.

* SET COLON ON/OFF (default is ON) affects whether GET variables are bounded by colons.

* SET DEBUG ON/OFF (default is OFF) allows easier debugging of "@" commands by shifting ECHO and STEP messages to the printer.

* SET SCREEN ON/OFF (default is ON) allows use of full screen operations.

* SET FORMAT TO SCREEN/PRINT/<format file> determines device destination of output (SCREEN or PRINTer). SET FORMAT TO <format file> establishes a format file as the source of "@" commands for the READ command. SCREEN is the default value.

* READ enters the editing mode so that GET variables can be altered.

ACCEPT

ACCEPT ["<cstring>"] TO <memvar>

This construct permits the entry of character strings into memory variables just as the INPUT command, but without the necessity of enclosing them in the quote marks required by the INPUT command. ACCEPT makes a memory variable of the type 'character' out of whatever is entered; INPUT determines the data type from the syntax of the entry and makes a memory variable of that type.

The <memvar> is created, if necessary, and the input character string is stored into <memvar>. If "<cstring>" is present, it is displayed on the screen, followed by a colon, as a prompt message before the input is accepted. If a carriage return is entered in response to an ACCEPT request, <memvar> will receive a single space character. Either single quotes, double quotes, or square brackets may be used to delimit the prompt string, however, both the beginning and ending marks must correspond.

Examples:

```
. ACCEPT "ENTER PERSONS NAME" TO NAM
ENTER PERSONS NAME:John Jones

. ACCEPT "ENTER PERSON'S NAME" TO NAM2
ENTER PERSON'S NAME:Dave Smith

. DISP MEMO
NAM          (C)  John Jones
NAM2         (C)  Dave Smith
** TOTAL **        02 VARIABLES USED  00020 BYTES USED

. ACCEPT TO ANY
 :ANY CHARACTERS

. DISP MEMO
NAM              John Jones
NAM2             Dave Smith
ANY              ANY CHARACTERS
** TOTAL **        03 VARIABLES USED  00034 BYTES USED
```

APPEND

```
a. APPEND FROM <file> [FOR <exp>] [SDF] [DELIMITED WITH <delimiter>]
b. APPEND BLANK
c. APPEND
```

In all three forms, records are appended onto the database in USE. APPEND, CREATE, and INSERT are the only commands that allow the addition of records to a database. APPEND and CREATE allow multiple additions at one time, INSERT allows only one.

In the first form, the records to be appended are taken from another file, i.e. <file>. If the SDF clause is present, the records are assumed to be in System Data Format (see section 6.0). If the new records are smaller than the old records in the USE file, then the new record is padded on the right side with blanks; if the new records are longer then the USE file records, then the newly appended records are truncated. Records are added to the USE file until end-of-file is detected upon the FROM file.

If the DELIMITED keyword is in the APPEND command, then the records taken from the FROM file are assumed to be delimited and appended accordingly. Many computer languages generate files where character strings are enclosed in single or double quotes and fields are separated by commas. In the delimited mode, dBASE removes the quotes and commas from delimited files and stores the data into a dBASE-structured database, according to the database's structure.

If the SDF and DELIMITED clauses are not present, then the FROM file is assumed to be a dBASE-structured database file. The structures of the USE and FROM file are compared. Fields which occur in the records of both files are taken from the FROM file and appended onto the USE file. Padding and truncation are performed as appropriate to force the FROM data items into the USE file's structure.

If the FOR phrase is used, then dBASE appends the records in the FROM <file> one by one, each time checking to see if the condition in the FOR is true. That is, the first record is appended. If the expression is true then the record is kept and dBASE will skip on to the next record. If the expression ~~then the~~ is False then the record is discarded and dBASE will again skip on to the next record. This procedure will continue until the end-of-file is reached for the FROM <file>. The implications of this is that the fields used in the expression must reside in the file <u>receiving</u> the new records.

If the BLANK clause (form b) is specified, a single, space filled record is appended to the USE file. This record can then be filled by the EDIT or REPLACE statements.

If no clauses follow the APPEND command (form c.), the user is prompted with the field names from the USE file's structure. Any number of new records may be created from the keyboard. The append mode is terminated when a carriage return is entered as the first character of the first field.

If the database in USE is an indexed database then the index file specified in the USE command is automatically updated when the new records are appended (except for APPEND BLANKs). Any other index file associated with that database must be re-indexed.

When APPENDing in the full-screen mode, the SET CARRY ON command will cause all of the data from the previous record to be carried over to the next record. Changes can then be made. This is especially useful if sucessive records have a lot of common data.

The APPEND command is especially useful when it is necessary to expand/contract fields or add/delete fields from an existing database. Using the CREATE command, set up a new database containing the desired structure and then APPEND the old database to the new. Fields which appear only in the new database will be blank filled.

Examples:

```
. USE EXAMPLE

. DISPLAY STRUCTURE
STRUCTURE FOR FILE:  EXAMPLE
NUMBER OF RECORDS:   00005
DATE OF LAST UPDATE: 12/31/80
PRIMARY USE DATABASE
FLD        NAME      TYPE WIDTH   DEC
001       NAME         C    020
002       TELE:EXTSN   C    005
003       MAIL:STOP    C    010
** TOTAL **               00036

. DISPLAY ALL
00001  NEUMAN, ALFRED E.      1357  123/456
00002  RODGERS, ROY           2468  180/103
00003  CASSIDY, BUTCH         3344  264/401
00004  CHANG, LEE             6743  190/901
00005  POST, WILEY            1011  84/13B
```

```
. APPEND

RECORD 00006

NAME:         LANCASTER, WILLIAM J
TELE:EXTSN:  6623
MAIL:STOP:   170/430

RECORD 00007

NAME:         NORRIS, R. "BOB"
TELE:EXTSN:  8093
MAIL:STOP:   427/396

RECORD 00008

NAME:        (cr)


. DISPLAY ALL OFF NAME,TELE:EXTSN
NEUMAN, ALFRED E.     1357
RODGERS, ROY          2468
CASSIDY, BUTCH        3344
CHANG, LEE            6743
POST, WILEY           1011
LANCASTER, WILLIAM J 6623
NORRIS, R. "BOB"      8093

  APPEND FROM DUPE3
00007 RECORDS ADDED

  DISPLAY ALL
00001  NEUMAN, ALFRED E.     1357  123/456
00002  RODGERS, ROY          2468  180/103
00003  CASSIDY, BUTCH        3344  264/401
00004  CHANG, LEE            6743  190/901
00005  POST, WILEY           1011  84/13B
00006  LANCASTER, WILLIAM J 6623  170/430
00007  NORRIS, R. "BOB"      8093  427/396
00008  NEUMAN, ALFRED E.     1357
00009  RODGERS, ROY          2468
00010  CASSIDY, BUTCH        3344
00011  CHANG, LEE            6743
00012  POST, WILEY           1011
00013  LANCASTER, WILLIAM J 6623
00014  NORRIS, R. "BOB"      8093

  APPEND BLANK

  DISPLAY
00015

  REPLACE NAME WITH 'RINEHART, RALPH
00001 REPLACEMENT(S)
```

```
. DISPLAY
00015  RINEHART, RALPH

. DISPLAY ALL NAME,' ex =',TELE:EXTSN
00001  NEUMAN, ALFRED E.        ex = 1357
00002  RODGERS, ROY             ex = 2468
00003  CASSIDY, BUTCH           ex = 3344
00004  CHANG, LEE               ex = 6743
00005  POST, WILEY              ex = 1011
00006  LANCASTER, WILLIAM J     ex = 6623
00007  NORRIS, R. "BOB"         ex = 8093
00008  NEUMAN, ALFRED E.        ex = 1357
00009  RODGERS, ROY             ex = 2468
00010  CASSIDY, BUTCH           ex = 3344
00011  CHANG, LEE               ex = 6743
00012  POST, WILEY              ex = 1011
00013  LANCASTER, WILLIAM J     ex = 6623
00014  NORRIS, R. "BOB"         ex = 8093
00015  RINEHART, RALPH          ex =


. USE B:SHOPLIST

. DISP STRU
STRUCTURE FOR FILE:    B:SHOPLIST.DBF
NUMBER OF RECORDS:     00009
DATE OF LAST UPDATE:   06/22/79
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH    DEC
001        ITEM        C     020
002        NO          N     005
003        COST        N     010    002
** TOTAL **                00036


. CREATE
FILENAME: NEWSHOP
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD     NAME,TYPE,WIDTH,DECIMAL PLACES
 001       ITEM,C,25
 002       NO,N,5
 003       COST,N,10,2
 004       NEED:DATE,C,8
 005       (cr)
INPUT NOW? N

  USE NEWSHOP
```

```
. APPEND FROM B:SHOPLIST
00009 RECORDS ADDED

. LIST
00001  BEANS                              5       0.75
00002  BREAD LOAVES                       2       0.97
00003  T-BONE                             4       3.94
00004  PAPER PLATES                       1       0.86
00005  PLASTIC FORKS                      5       0.42
00006  LETTUCE                            2       0.53
00007  BLEU CHEESE                        1       1.96
00008  MILK                               2       1.30
00009  CHARCOAL                           2       0.75

. REPLACE ALL NEED:DATE WITH ' 7/ 4/76'
00009 REPLACEMENT(S)

. LIST
00001  BEANS                              5       0.75  7/ 4/76
00002  BREAD LOAVES                       2       0.97  7/ 4/76
00003  T-BONE                             4       3.94  7/ 4/76
00004  PAPER PLATES                       1       0.86  7/ 4/76
00005  PLASTIC FORKS                      5       0.42  7/ 4/76
00006  LETTUCE                            2       0.53  7/ 4/76
00007  BLEU CHEESE                        1       1.96  7/ 4/76
00008  MILK                               2       1.30  7/ 4/76
00009  CHARCOAL                           2       0.75  7/ 4/76
```

(The following example demonstrates the DELIMITED file append. This file could have been created by a number of different versions of BASIC)

```
'BARNETT, WALT',31415,6
'NICHOLS, BILL',76767,17
'MURRAY, CAROL',89793,4
'WARD, CHARLES A.',92653,15
'ANDERSON, JAMES REGINALD III','11528',   16
```

(Append the file into a dBASE-structured database)

```
. USE ORDERS

. DISP STRU
STRUCTURE FOR FILE:    ORDERS.DBF
NUMBER OF RECORDS:     00008
DATE OF LAST UPDATE:   00/00/00
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001       CUSTOMER     C    020
002       PART:NO      C    005
003       AMOUNT       N    005
** TOTAL **                00031

. LIST
00001  SWARTZ, JOE            31415     13
```

```
00002  SWARTZ, JOE            76767     13
00003  HARRIS, ARNOLD         11528     44
00004  ADAMS, JEAN            89793     12
00005  MACK, JAY              31415      3
00006  TERRY, HANS            76767      5
00007  JUAN, DON              21828      5
00008  SALT, CLARA            70296      9

. APPEND FROM DELIM.DAT DELIMITED
00005 RECORDS ADDED

. LIST
00001  SWARTZ, JOE            31415     13
00002  SWARTZ, JOE            76767     13
00003  HARRIS, ARNOLD         11528     44
00004  ADAMS, JEAN            89793     12
00005  MACK, JAY              31415      3
00006  TERRY, HANS            76767      5
00007  JUAN, DON              21828      5
00008  SALT, CLARA            70296      9
00009  BARNETT, WALT          31415      6
00010  NICHOLS, BILL          76767      7
00011  MURRAY, CAROL          89793      4
00012  WARD, CHARLES A.       92653     15
00013  ANDERSON, JAMES REGI   11528     16
```

(The following examples demonstrates an APPEND FROM <file> FOR <exp>. Note that the fields in the FOR are in the USE file also.)

```
. USE CHECKS
. DISP STRU
STRUCTURE FOR FILE:  CHECKS.DBF
NUMBER OF RECORDS:   00013
DATE OF LAST UPDATE: 10/18/81
PRIMARY USE DATABASE
FLD        NAME        TYPE WIDTH   DEC
001        NUMBER       N     005
002        RECIPIENT    C     020
003        AMOUNT       N     010   002
004        HOME         L     001
005        OUTGOING     L     001
** TOTAL **                 00038
```

```
. LIST
00001     1 Phone Company              104.89 .F. .T.
00002     2 Gas Company                  4.14 .F. .T.
00003     3 Electricity                250.31 .F. .T.
00004     4 Grocery Store             1034.45 .F. .T.
00005    34 Me                         561.77 .T. .F.
00006     6 Bank, service charge         4.00 .T. .T.
00007     7 Doctor Doolittle           100.00 .T. .T.
00008     8 Pirates                    101.01 .F. .T.
00009     9 Car Repair Man             500.01 .F. .T.
00010    10 Me                         561.01 .T. .F.
00011    11 Tuperware                   50.02 .F. .T.
00012    12 Me                         561.77 .T. .F.
00013    13 Me                         750.03 .T. .F.

. USE MONTH
. DISP STRU
STRUCTURE FOR FILE:  MONTH.DBF
NUMBER OF RECORDS:   00003
DATE OF LAST UPDATE: 10/18/81
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001      NUMBER       N     005
002      AMOUNT       N     010   002
003      HOME         L     001
** TOTAL **               00017

. LIST
00001      29        14.89 .T.
00002      16       764.09 .T.
00003      78        97.96 .T.

. APPEND FROM CHECKS FOR HOME
00006 RECORDS ADDED

. APPEND FROM CHECKS FOR OUTGOING
*** SYNTAX ERROR ***
                              ?
APPEND FROM CHECKS FOR OUTGOING
CORRECT AND RETRY(Y/N)? N
```

That last append was to show what would happen if the FOR field was not in the USE file.

BROWSE

BROWSE

The BROWSE command is one of the most powerful dBASE commands for data editing and viewing. The data from up to 19 records is displayed onto the screen (fewer if fields are greater than 80 characters). As many fields as will fit are put on each line. The screen should be considered as a window into a database. You can scroll backwards and forwards through the records and you can pan left and right through the fields of the database. Any data can be edited with the standard full-screen editing method (see section 8 for additional information)

This is a summary of the full-screen control keys that will work in BROWSE:

ctl-E,A	-	backs up to the previous data field;
ctl-X,F	-	advances to the next data field;
ctl-D	-	advances to the next character;
ctl-S	-	backs up to the last character;
ctl-G	-	deletes the character under the cursor;
RUBOUT	-	deletes the character before the cursor;
ctl-Q	-	exits without saving the changes;
ctl-W	-	exits and saves the changes (ctl-O for Superbrain);
ctl-B	-	pans the window left one field;
ctl-Z	-	pans the window right one field;
ctl-C	-	writes the current record and advances one record;
ctl-R	-	writes the current record and backs up one record;
ctl-U	-	switches (toggles) the current record between being marked for deletion and not being marked.

Example:

```
BROWSE
```

CANCEL

CANCEL

Cancel a command file execution and return to the normal keyboard interpretive mode.

Example:

```
INPUT 'IS JOB DONE (Y/N)' TO X
IF X
  CANCEL
ENDIF
```

This is a fragment from a command file. The INPUT command asks for a yes/no answer. If the answer is yes ('Y', 'y', 'T', or 't') then the IF X line of the command file will be satisfied (since X will be logically .TRUE.) and the CANCEL command will be executed.

See Appendix A for more examples.

CHANGE

```
CHANGE [<scope>] FIELD <list> [FOR <exp>]
```

CHANGE is a command that allows the user to make a number of alterations to a database with minimum effort. All database fields that are referenced in the list are presented to the user in the order given by <list>. The user has the opportunity of entering new data, modifing the data or skipping to the next field. When the <list> has been exhausted, CHANGE will proceed to the next record as specified in the <scope>. The default scope is the current record.

A field can be del[illegible]d in its entirety by typing a control-Y (followed by a retur[illegible] in response to the CHANGE? message. The CHANGE command can be [illegible]orted by typing an ESCAPE character.

Example:

```
. USE CARDS
. CHANGE FIELD DATE

RECORD: 00001

DATE: 08/19/81
CHANGE? 81
TO        82

DATE: 08/19/82
CHANGE? (cr)
```

CLEAR

CLEAR [GETS]

If the GETS (or GET) keyword is used then all of the GETs that are pending (i.e. a GET set up by the @ command) are cleared and the screen is left intact. This is opposed to the ERASE command which also clears pending GETs and also erases the screen.

If there is no GETS keyword, then this command resets dBASE II. All databases in USE are closed and un-used, all memory variables are released, and the PRIMARY work area is re-selected.

This command gives dBASE II a "clean slate". For instance: if a command file finished executing and left dBASE in the SECONDARY state, then executing a new command file that assumes that the PRIMARY state was selected, will cause unknown things to happen.

CLEAR should be used at the beginning of a command file to give the command file a known state.

Example:

```
    CLEAR
```

CONTINUE

This command is used with the LOCATE command. LOCATE and CONTINUE may be separated by other commands, however there are limitations. See the LOCATE command for more information.

COPY

```
COPY TO <file> [<scope>] [FIELD <list>] [FOR <exp>]
      [SDF] [STRUCTURE] [DELIMITED [WITH <delimiter>]]
```

This command copies the database in USE to another file. The <file> may be in dBASE format or in the System Data Format (if the SDF option is specified).

If the STRUCTURE clause is specified, then only the structure of a dBASE file in USE is copied to the "TO" file.

If a list of fields is supplied following a FIELD clause, then only those data fields are copied TO the file. For the COPY STRUCTURE FIELD <list>, only the structure of the listed fields is copied TO the file. In either case, the new structure will be made up of only those fields specified by the FIELD clause. No FIELD clause specifies that all fields will be copied.

If the SDF clause is specified, then the file in USE is copied to another file without the structure. This new file will be in ASCII standard format. This allows the generation of files which can be input to processors other than dBASE. The STRUCTURE and SDF clauses are mutually exclusive.

If the DELIMITED keyword is also in the command, then the output file will have all of its character string type fields enclosed in quotes and the fields will be separated by commas. This is the converse of a delimited APPEND. By default, the DELIMITED type of COPY uses single quotes as delimiters to mark character string fields. The WITH sub-phrase of the DELIMITED phrase allows any character to be the delimiter. If a "," is used as the delimiter then the character fields will have trailing blanks trimmed, the numeric fields will have the leading blanks trimmed, and the character strings will not be enclosed in quotes. The APPEND command will only respond to single and double quotes.

If either the DELIMITED or SDF option is used then the output <file> name will default to a .TXT extension, otherwise the output file will default to a .DBF extension.

The "TO" file is created if it does not exist.

Examples:

```
. DISPLAY ALL OFF NAME,TELE:EXTSN
NEUMAN, ALFRED E.     1357
RODGERS, ROY          2468
CASSIDY, BUTCH        3344
CHANG, LEE            6743
POST, WILEY           1011
LANCASTER, WILLIAM J  6623
NORRIS, R. "BOB"      8093

. DISPLAY STRUCTURE
STRUCTURE FOR FILE:   EXAMPLE
NUMBER OF RECORDS:    00007
DATE OF LAST UPDATE:  00/00/00
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001     NAME          C    020
002     TELE:EXTSN    C    005
003     MAIL:STOP     C    010
** TOTAL **               00036

. COPY TO DUPE
00007 RECORDS COPIED

. COPY TO DUPE2 FOR TELE:EXTSN<'8000'
00006 RECORDS COPIED

. USE DUPE2

. DISPLAY ALL
00001  NEUMAN, ALFRED E.     1357  123/456
00002  RODGERS, ROY          2468  180/103
00003  CASSIDY, BUTCH        3344  264/401
00004  CHANG, LEE            6743  190/901
00005  POST, WILEY           1011  84/13B
00006  LANCASTER, WILLIAM J  6623  170/430

. USE EXAMPLE

. COPY FIELD NAME,TELE:EXTSN TO DUPE3
00007 RECORDS COPIED

. USE DUPE3
```

```
STRUCTURE FOR FILE:   DUPE3
NUMBER OF RECORDS:    00007
DATE OF LAST UPDATE:  00/00/00
PRIMARY USE DATABASE
FLD        NAME        TYPE WIDTH   DEC
001      NAME           C    020
002      TELE:EXTSN     C    005
** TOTAL **               00036

. DISPLAY ALL
00001  NEUMAN, ALFRED E.     1357
00002  RODGERS, ROY          2468
00003  CASSIDY, BUTCH        3344
00004  CHANG, LEE            6743
00005  POST, WILEY           1011
00006  LANCASTER, WILLIAM J  6623
00007  NORRIS, R. "BOB".     8093
. USE EXAMPLE

. COPY NEXT 4 TO DUPE5
00004 RECORDS COPIED

. USE DUPE5

. DISPLAY ALL
00001  NEUMAN, ALFRED E.     1357  123/456
00002  RODGERS, ROY          2468  180/103
00003  CASSIDY, BUTCH        3344  264/401
00004  CHANG, LEE            6743  190/901
```

(The delimited COPY)

```
. USE ORDERS

. DISP STRUCTURE
STRUCTURE FOR FILE:   ORDERS.DBF
NUMBER OF RECORDS:    00012
DATE OF LAST UPDATE:  07/01/80
PRIMARY USE DATABASE
FLD        NAME        TYPE WIDTH   DEC
001      CUSTOMER       C    020
002      PART:NO        C    005
003      AMOUNT         N    005
** TOTAL **               00031
```

```
. LIST
00001  SWARTZ, JOE          31415   13
00002  SWARTZ, JOE          76767   13
00003  HARRIS, ARNOLD       11528   44
00004  ADAMS, JEAN          89793   12
00005  MACK, JAY            31415    3
00006  TERRY, HANS          76767    5
00007  JUAN, DON            21828    5
00008  SALT, CLARA          70296    9
00009  BARNETT, WALT        31415    6
00010  NICHOLS, BILL        76767   17
00011  MURRAY, CAROL        89793    4
00012  WARD, CHARLES A.     92653   15

. COPY TO DELIM.DAT DELIMITED
00012 RECORDS COPIED

'SWARTZ, JOE           ','31415',   13
'SWARTZ, JOE           ','76767',   13
'HARRIS, ARNOLD        ','11528',   44
'ADAMS, JEAN           ','89793',   12
'MACK, JAY             ','31415',    3
'TERRY, HANS           ','76767',    5
'JUAN, DON             ','21828',    5
'SALT, CLARA           ','70296',    9
'BARNETT, WALT         ','31415',    6
'NICHOLS, BILL         ','76767',   17
'MURRAY, CAROL         ','89793',    4
'WARD, CHARLES A.      ','92653',   15
```

COUNT

COUNT [<scope>] [FOR <exp>] [TO <memvar>]

Count the number of records in the USE file. If the FOR clause is invoked, then only the number of records which satisfy the expression are counted. If the TO clause is included, the integer count is places into a memory variable. The memory variable will be created if it did not exist prior to this command.

dBASE responds with the message:

```
  COUNT = xxxxx
```

Examples:

```
. USE INVNTRY

. DISPLAY STRUCTURE
STRUCTURE FOR FILE:   INVNTRY
NUMBER OF RECORDS:    00010
DATE OF LAST UPDATE:  10/23/78
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001     ITEM:NO       N    006
002     CLASS:NO      N    003
003     VENDOR:NO     N    005
004     DESCR         C    013
005     UNIT:COST     N    007    002
006     LOCATION      C    005
007     ON:HAND       N    004
008     SOLD          N    004
009     PRICE         N    007    002
** TOTAL **                00055

. DISPLAY ALL
00001  136928  13  1673 ADJ. WRENCH      7.13 189      9    0     9.98
00002  221679   9  1673 SM. HAND SAW     5.17 173      4    1     7.98
00003  234561   0    96 PLASTIC ROD      2.18 27     112   53     4.75
00004  556178   2   873 ADJ. PULLEY     22.19 117      3    0    28.50
00005  723756  73    27 ELECT.BOX       19.56 354      6    1    29.66
00006  745336  13    27 FUSE BLOCK      12.65 63       7    2    15.95
00007  812763   2  1673 GLOBE            5.88 112      5    2     7.49
00008  876512   2   873 WIRE MESH        3.18 45       7    3     4.25
00009  915332   2  1673 FILE             1.32 97       7    3     1.98
00010  973328   0    27 CAN COVER        0.73 21      17    5     0.99

. COUNT
COUNT = 00010


. COUNT FOR ITEM:NO>500000
COUNT = 00007
```

```
. COUNT FOR 'ADJ'$DESCR
COUNT = 00002

. GOTO TOP

. COUNT FOR PRICE<10 NEXT 6
COUNT = 00003

. GOTO TOP

. COUNT NEXT 6 FOR PRICE<10
COUNT = 00003

. USE B:SHOPLIST

. LIST
00001  BEANS                5       0.75
00002  BREAD LOAVES         2       0.97
00003  T-BONE               4       3.94
00004  PAPER PLATES         1       0.86
00005  PLASTIC FORKS        5       0.42
00006  LETTUCE              2       0.53
00007  BLEU CHEESE          1       1.96
00008  MILK                 2       1.30
00009  CHARCOAL             2       0.75

. DISPLAY STRUCTURE
STRUCTURE FOR FILE:    B:SHOPLIST.DBF
NUMBER OF RECORDS:     00009
DATE OF LAST UPDATE:   12/10/76
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001       ITEM         C     020
002       NO           N     005
003       COST         N     010   002
** TOTAL **                00036

. COUNT TO XX FOR COST>1
COUNT = 00003

. ? XX
  3
```

CREATE

```
CREATE [<filename>]
```

A new dBASE structured file is CREATEd. The user provides the structure, field names, and file name for the database file.

If not supplied in the command, the user is first prompted for the <filename> to be used by the message:

```
FILENAME:
```

The user enters a valid filename with the following added restriction: the filename may contain no special characters other than those normally used by CP/M for special purposes (such as B: to denote disk drive "B").

If the file existed before the create command was given, dBASE asks the user:

DESTROY EXISTING FILE? To which the user must reply Y or N as the case may be.

If the file is new to the system or if the user answered Y to the destroy question, dBASE is now ready to accept the structure of the data base from the user. The following message is displayed:

```
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD    NAME,TYPE,WIDTH,DECIMAL PLACES
 001
```

The user now enters field names and associated structure information. A field name is a character string up to 10 characters long which consists of alphabetic letters, numeric digits, and colons. Field names must begin with an alphabetic character. Fields may be any of three types: character string, numeric, or logical. The type field is specified by one character, as:

C - character string
N - numeric
L - logical

The width refers to the length of the field, for instance, a character string may be 20 characters long i.e. it's width is 20. Numeric data may be either integer or decimal. The width of integers is the maximum number of digits that they may be expected to contain. For decimal numbers, two widths are required; the first is the maximum number of digits that the decimal number is expected to contain (including the decimal point), the second width is the number of digits which are to by allowed on the right side of the decimal point. Logical data may only be of length 1.

Examples:

```
. CREATE
FILENAME:EXAMPLE
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD     NAME,TYPE,WIDTH,DECIMAL PLACES
 001      NAME,C,20
 002      TELE:EXTSN,C,5
 003      MAIL:STOP,C,10
 004      (cr)
INPUT NOW?Y

RECORD 00001

NAME:         NEUMAN, ALFRED E.
TELE:EXTSN:  1357
MAIL:STOP:   123/456

RECORD 00002

NAME:         RODGERS, ROY
TELE:EXTSN:  2468
MAIL:STOP:   180/103

RECORD 00003

NAME:         CASSIDY, BUTCH
TELE:EXTSN:  3344
MAIL:STOP:   264/401

RECORD 00004

NAME:         CHANG, LEE
TELE:EXTSN:  6743
MAIL:STOP:   190/901
```

```
RECORD 00005

NAME:         POST, WILEY
TELE:EXTSN:  1011
MAIL:STOP:   84/13B

RECORD 00006

NAME:         (cr)

. DISPLAY STRUCTURE
NO FILE IN USE, FILENAME:  EXAMPLE
STRUCTURE FOR FILE:   EXAMPLE
NUMBER OF RECORDS:    00005
DATE OF LAST UPDATE:  00/00/00
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001      NAME          C     020
002      TELE:EXTSN    C     005
003      MAIL:STOP     C     010
** TOTAL **                00036

  DISPLAY ALL
00001  NEUMAN, ALFRED E.     1357   123/456
00002  RODGERS, ROY          2468   180/103
00003  CASSIDY, BUTCH        3344   264/401
00004  CHANG, LEE            6743   190/901
00005  POST, WILEY           1011   84/13B
```

DELETE

```
DELETE [<scope>] [FOR <exp>]
DELETE FILE <filename>
```

All records which are within <scope> (and which satisfy the FOR expression if present) are marked for deletion. The default scope is the current record only. Records are not physically deleted until a PACK operation, however records marked for deletion will not be copied, appended, or sorted. The RECALL operation may be used to revive records marked as deleted. Records which are marked for deletion can be displayed. The mark of deletion appears as an asterisk between the record number and the first field.

In the second form, the file named <filename> will be removed from the disk drive where it resides (if possible) and the space it was occupying will be released to the operating system for reassignment. If, however, the <filename> is currently in use, the file will not be deleted.

Examples:

```
 LIST
00001  136928  13  1673 ADJ. WRENCH       7.13 189    9    0    9.98
00002  221679   9  1673 SM. HAND SAW      5.17 173    4    1    7.98
00003  234561   0    96 PLASTIC ROD       2.18 27   112   53    4.75
00004  556178   2   873 ADJ. PULLEY      22.19 117    3    0   28.50
00005  723756  73    27 ELECT.BOX        19.56 354    6    1   29.66
00006  745336  13    27 FUSE BLOCK       12.65 63     7    2   15.95
00007  812763   2  1673 GLOBE             5.88 112    5    2    7.49
00008  876512   2   873 WIRE MESH         3.18 45     7    3    4.25
00009  915332   2  1673 FILE              1.32 97     7    3    1.98

. DELETE RECORD 2
00001 DELETION(S)

. 5

. DELETE NEXT 3
00003 DELETION(S)
```

```
. LIST
00001  136928  13  1673 ADJ. WRENCH        7.13 189       9    0     9.98
00002 *221679   9  1673 SM. HAND SAW       5.17 173       4    1     7.98
00003  234561   0    96 PLASTIC ROD        2.18 27      112   53     4.75
00004  556178   2   873 ADJ. PULLEY       22.19 117       3    0    28.50
00005 *723756  73    27 ELECT.BOX         19.56 354       6    1    29.66
00006 *745336  13    27 FUSE BLOCK        12.65 63        7    2    15.95
00007 *812763   2  1673 GLOBE              5.88 112       5    2     7.49
00008  876512   2   873 WIRE MESH          3.18 45        7    3     4.25
00009  915332   2  1673 FILE               1.32 97        7    3     1.98

. RECALL ALL
00004 RECALL(S)

. LIST
00001  136928  13  1673 ADJ. WRENCH        7.13 189       9    0     9.98
00002  221679   9  1673 SM. HAND SAW       5.17 173       4    1     7.98
00003  234561   0    96 PLASTIC ROD        2.18 27      112   53     4.75
00004  556178   2   873 ADJ. PULLEY       22.19 117       3    0    28.50
00005  723756  73    27 ELECT.BOX         19.56 354       6    1    29.66
00006  745336  13    27 FUSE BLOCK        12.65 63        7    2    15.95
00007  812763   2  1673 GLOBE              5.88 112       5    2     7.49
00008  876512   2   873 WIRE MESH          3.18 45        7    3     4.25
00009  915332   2  1673 FILE               1.32 97        7    3     1.98

. DISP FILES ON B
DATABASE FILES    # RCDS     LAST UPDATE
SHOPLIST          00007      06/06/76
SHOPSAVE          00007      06/05/76

. DELETE FILE B:SHOPSAVE
FILE DELETED

. DISPLAY FILES ON B
DATABASE FILES    # RCDS     LAST UPDATE
SHOPLIST          00007      06/06/76
```

DISPLAY

```
a. DISPLAY [<scope>] [FOR <exp>] [<exp list>] [OFF]
b. DISPLAY STRUCTURE
c. DISPLAY MEMORY
d. DISPLAY FILES [ON <disk drive>] [LIKE <skeleton>]
```

Display is the foundation of dBASE. The end goal of all database operation is to display the data in the database (or cross sections and abstractions of the data) upon demand. DISPLAY satisfies that goal by allowing a wide variety of forms that select the wanted data.

In case a. all or part of the database in USE is displayed. If <scope> is not specified and the FOR <exp> is not in the command, only the current record can contribute information for display. If <scope> is not specified and there is a FOR <exp>, then all records in the database may contribute to the display. All fields are displayed unless the <exp list> clause is specified. Valid expressions may consist of data fields, memory variables, or any valid literal number, character or logical. The current record number is prefixed to each line displayed unless the OFF option is selected. If the FOR clause is specified, then only those records that satisfy the FOR's conditional expression can contribute information for display.

After groups of 15 records have been displayed, DISPLAY waits for any keystroke to continue. This allows the user to "page" through a long display. The LIST command is identical to the DISPLAY command except that LIST does not wait after record groups and it's default scope is ALL records. An ESCape character terminates the DISPLAY or LIST commands.

In case b. only the structure of the database in USE is displayed.

In case c. all currently defined memory variables are displayed as memory variable name and associated value.

Case d. is a way to display .DBF files that are residing on the default unit (or on <disk drive>) along with some of the database's statistics. The LIKE phrase allows other types of files to be displayed. The <skeleton> is usually of the form *.type, where type is TXT, FRM, MEM, or any other three letter string. These files are displayed just as in the CP/M DIR command.

Examples:

```
. USE B:INVENTRY

. DISPLAY STRUCTURE
STRUCTURE FOR FILE:  B:INVENTRY.DBF
NUMBER OF RECORDS:   00008
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH   DEC
001       ITEM          C    020
002       COST          N    010    002
003       PART:NO       C    005
004       ON:HAND       N    005
** TOTAL                   00041 BYTES     (note: total includes
                                            1 overhead byte)

. DISPLAY ALL ITEM, PART:NO, COST*ON:HAND ,$(PART:NO,1,2) FOR ;
 COST > 100 .AND. ON:HAND > 2 OFF
TANK, SHERMAN          89793          404997.00  89
TROMBONES              76767           15076.12  76
RINGS, GOLDEN          70296            1000.00  70

. DISPLAY MEMORY
CLIENT:NAM (C)  DANGLEMEYER, PRENTICE
BUDGET     (N)   123456.70
GF:STATUS  (L)  .T.
** TOTAL **       03 VARIABLES USED  00027 BYTES USED

. DISPLAY FILES ON B: LIKE *.FRM
TEST      FRM      ADMIN     FRM      ORDERS    FRM

. DISPLAY FILES
DATABASE FILES  #RCDS   LAST UPDATE
TEST     DBF    00077   00/00/00
ADRECS   DBF    00073   09/23/81
HISTSTR  DBF    00000   06/29/81
TMPADMIN DBF
NOT A dBASE II DATABASE
```

he last .DBF file in the list above is the file that is not the
BASE database.

nly representative examples of DISPLAY are given here, refer to
ther commands for other examples.

DO

```
a. DO <file>
b. DO WHILE <exp>
   <statements>
   ENDDO
c. DO CASE
     CASE <exp>
        <statements>
     CASE <exp>
        <statements>
     .
     .
     .
     [OTHERWISE]
        <statements>
   ENDCASE
```

In case a, <file> is opened and read. The file in this case is known as a COMMAND FILE. It consists entirely of dBASE commands. The input is interpreted and executed as keyboard commands are. DO's can be stacked up to 16 deep (i.e. command files can contain DO commands which invoke other command files). Control is released by a command file with an end-of-file or by the RETURN command. If the current command file was called by a command file, control will be given back to the higher level command file. If, during the execution of a command file, a CANCEL command is encountered, all command files are closed and the keyboard is made the source for future commands.

In case b, if the <exp> evaluates as a logical TRUE, the statements following the DO are executed until an ENDDO statement is encountered. If the <exp> evaluates to a logical FALSE, control is transferred to the statement following the ENDDO statement.

Note: <statements> refers to entire statements. The DO WHILE statement ends with an ENDDO. Statements must nest properly; if there is an IF "inside" a DO WHILE, then an ENDDO may not occur before the ENDIF. See section 9.2 Rule 8 for more information.

Examples:

```
DO ACCNTPAY

DO WHILE .NOT.EOF
  DISPLAY NAME
  .
  .
  .
  SKIP
ENDDO
```

CASE is an extension of the DO command and takes the form shown above. There is no limit to the number of CASE phrases that a DO CASE may contain. The OTHERWISE phrase is optional.

DO CASE is a structured procedure. The individual CASEs in the construct could be viewed as the exceptions to the rule that defines the OTHERWISE. If some condition needs some special processing then the condition would be a CASE and all other conditions would be the OTHERWISE. OTHERWISE may also be viewed as the default condition. See the first example below.

How dBASE handles the DO CASE construct may best be explained as a series of IFs. That is, dBASE will execute the DO CASE as if it were a list of IF-ENDIFs.

```
DO CASE                                      IF ITEM='ORANGES'
   CASE ITEM='ORANGES'                          any statements
     any statements                          ELSE
   CASE ITEM='APPLES'           =               IF ITEM='APPLES'
     any statements                                any statements
   OTHERWISE                                    ELSE
     any statements                                any statements
   ENDCASE                                      ENDIF
                                             ENDIF
```

Thus, dBASE will examine the <exp>s in the individual CASEs and the first one that is true will have the statements after it executed. When dBASE reaches the next phrase beginning with a "CASE" it will exit to the ENDCASE. This means that if more than one CASE is true, only the first one will be executed.

If the OTHERWISE clause is present and none of the CASEs are true, then the <statements> in the OTHERWISE clause will be executed. If there is no OTHERWISE clause and none of the CASEs are true, then the DO CASE will be exited with none of the <statements> executed at all.

Any statements that are placed between the "DO CASE" and the first "CASE" will not be executed.

Examples:

```
DO CASE
   CASE ITEM = "BROWN"
        <statements> that process BROWN
   CASE ITEM = "JONES"
        <statements> that process JONES
   CASE ITEM = "SMITH"
        <statements> that process SMITH
   OTHERWISE
        <statements> that process all the other names
ENDCASE
```

In the case above all the expressions were for the same field name. This is not necessary. An <exp> may contain anything and the series of CASEs need not have a tight relationship.

```
DO CASE
   CASE TODAY = "MONDAY"
        <statements> for MONDAY
   CASE WEATHER = "RAIN"
        <statements> for RAIN
   CASE CITY = "LOS ANGELES"
        <statements> for LOS ANGELES
ENDCASE
```

Of course, if it is a rainy Monday in Los Angeles only the CASE for MONDAY will be executed.

CASEs need not be all character strings as in these two examples. Any expression will work.

```
DO CASE
   CASE 3 = 2 + 1
        <statements> for addition
   CASE .NOT. A
        <statements> for boolean logic
   CASE "A"$"ABCDEF"
        <statements> for string logic
   OTHERWISE
        <statements>
ENDCASE
```

ENDCASE is the statement used to terminate a DO CASE structure. When a case or OTHERWISE has finished processing, control is resumed at the line following the ENDCASE.

EDIT

EDIT [n]

The EDIT command allows the user to selectively change the contents of the data fields in a database. Edit's usage and action varies, depending on whether on not dBASE is in the full-screen mode (see the SET SCREEN command).

When dBASE is in the full-screen mode, editing can be done by either "EDIT" or "EDIT n" (n represents the record to be edited). If n is not present then dBASE will ask for the coordinates of the record to be edited. This is similar to the non-full-screen mode, however, full-screen capabilities will still used after the record number is supplied. See section 8, full-screen operations, for a description of control keys and cursor movement.

When the edit command is used in the non-full-screen mode, dBASE responds with:

```
COORD:
```

The user then enters the coordinates of the data field to be changed and (optionally) the new value. The coordinates of the data field are: the record number, and the field number (or the field name). If a new value is supplied, dBASE will replace the contents of the specified field with the new value. If a new value is not supplied, dBASE displays the current value of the data field and prompts the user for changes. If no changes are desired, a carriage return will cause dBASE not to alter the contents of the field. Whether changes are made or not, dBASE will prompt the user for the next pair of coordinates with another "COORD:" message.

After the first set of coordinates have been entered, the user may omit either of the coordinate values and dBASE will use the previous value of that coordinate. The EDIT mode is exited by entering a carriage return as the response to the COORD request.

The entire data field can be erased by entering a control-Y, RETURN whenever the CHANGE? message is displayed. This permits a field to be completely reentered if desired. The editing of a data field can be aborted by entering a CTL-Q character. This discards any editing done and restores the data field to its original contents.

If an INDEXed file is being EDITed and the index clause was USEd, then dBASE will adjust the index if the key field is altered. If more than one index file is associated with the database, then the un-USEd files will be unaffected by the edit.
Examples:

```
    USE SHOPLIST
```

```
. DISPLAY STRUCTURE
STRUCTURE FOR FILE:   SHOPLIST
NUMBER OF RECORDS:    00006
DATE OF LAST UPDATE:  07/03/76
PRIMARY USE DATABASE
FLD        NAME      TYPE WIDTH   DEC
001       ITEM         C     020
002       NO           N     005
003       COST         N     010    002
** TOTAL **               00036

. LIST
00001  BEANS #303 CAN           5        0.69
00002  BREAD                    2        0.89
00003  T-BONE STEAKS            4        3.59
00004  LETTUCE                  1        0.49
00005  MILK (1 GAL BOTTLES)     2        1.19
00006  CHARCOAL                 1        0.69

. EDIT
COORD: 5,ITEM,MILK (1/2 GAL)

COORD: 2,1

ITEM: BREAD

CHANGE? D
TO      D LOAVES

ITEM: BREAD LOAVES
CHANGE? (cr)
COORD: 6,1

ITEM: CHARCOAL
CHANGE? AL
TO      AL, 5# BAGS

ITEM: CHARCOAL, 5# BAGS
CHANGE? (cr)
COORD: ,2

NO:      1
TO: 2
COORD: 4

NO:      1
TO: 2
COORD: (cr)
. LIST
00001  BEANS #303 CAN           5        0.69
00002  BREAD LOAVES             2        0.89
00003  T-BONE STEAKS            4        3.59
00004  LETTUCE                  2        0.49
00005  MILK (1/2 GAL)           2        1.19
```

```
 00006  CHARCOAL, 5# BAGS          2         0.69
```

(The following portion of a command file would also allow one to edit a database on a selective basis. The "&" is vital to making these commands work; it will change the string accepted by the ACCEPT into numbers that EDIT will recognize.)

```
STORE '1' TO X
DO WHILE X <> '0'
  ACCEPT "Enter Record Number" TO X
  EDIT &X
ENDDO
```

EJECT

EJECT

This command causes the printer to do a form feed (eject the page) if either PRINT is SET ON or FORMAT is SET TO PRINT. When using the @ command to do direct page formatting, the EJECT command also zeros the line and column registers.

Example:

```
. EJECT
```

ENDDO

The statement used to terminate a DO WHILE loop. When encountered, control is transferred back to the DO statement for re-assessment of the logical value of the <exp>.

See the DO command.

See Appendix A for examples.