RETURN

RETURN

This command is used inside a command file to return control to the command file which called it (or to the keyboard if the user called the command file directly). Encountering an end of file on a command file is equivalent to a RETURN command.

Command files usually have a RETURN command as their last executable line.

See Appendix A for examples.

SAVE

SAVE TO <file>

This command stores all currently defined memory variables to a file. These memory variables may be restored by the RESTORE command.

Examples:

```
. DISPLAY MEMORY
ONE         (N)   1.0000
ALFABET     (C)  ABCDEFGHIJKL
CHARS       (C)  ABCDEFGHIJKL NEW STUFF
** TOTAL **       03 VARIABLES USED  00042 BYTES USED

. SAVE TO MEMFILE

. RELEASE ALL

. DISPLAY MEMORY
** TOTAL **       00 VARIABLES USED  00000 BYTE USED

. RESTORE FROM MEMFILE

. DISPLAY MEMORY
ONE         (N)   1.0000
ALFABET     (C)  ABCDEFGHIJKL
CHARS       (C)  ABCDEFGHIJKL NEW STUFF
** TOTAL **       03 VARIABLES USED  00042 BYTES USED
```

SELECT

SELECT [PRIMARY]
[SECONDARY]

This command causes dBASE to select one of the two possible database areas for future operations. This permits the dBASE user to do operations on two databases at a time, such as using the data from one database to update the data in another database ,or comparing the data in two databases. or any of a number of other multi-database operations.

When dBASE is initiated, the PRIMARY area is active. PRIMARY will stay active until a SELECT SECONDARY instruction is given. The secondary area will then be active until a SELECT PRIMARY command is encountered. A different database may be USE'ed in each of the areas. This permits the (nearly) concurrent usage of two databases at once. There is no effect if a SELECT SECONDARY is entered when the secondary area is already selected or vice versa with the primary area.

When both database areas have databases in USE, field variables can be extracted from either area. That is to say, any expression can use variables from either database region. If the field names in both regions are the same for a desired variable, then the variable can be prefixed with a "P." or "S." to denote which database it is to come from.

dBASE commands that cause movement of the database (i.e. GOTO, SKIP, REPORT, SORT, COPY, LIST, DISPLAY (for a scope of more than one record), and others) affect only the currently selected database. The SET LINKAGE ON command will allow all sequential commands (those that have a <scope> parameter) perform positioning on both the secondary and the primary databases. (See the SET command). The REPLACE command will only affect variables in the currently selected database. The DISPLAY STRUCTURE command will display the structure of the currently selected database only.

Examples:

```
USE SHOPLIST
```

```
. LIST
00001  Beans                    5      0.75
00002  Bread loaves             2      1.06
00003  T-Bone steak             4      4.33
00004  Paper plates             1      0.94
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

. NOTE NOW OPEN ANOTHER DATABASE IN THE SECONDARY AREA

. SELECT SECONDARY

. USE SHOPCOST

. LIST
00001  800104       31.3
00002  800111       45.69
00003  800118       51.18
00004  800124       48.19
00005  800201       55.82
00006  800209       12.04
00007  800229       12.04

. SELECT PRIMARY

. SUM COST
 12.04

. SELECT SECONDARY

. APPEND

RECORD 00008

DATE       : 800303
AMOUNT     : 12.04

RECORD 00009

DATE       : (cr)

. SUM AMOUNT
 268.38

. NOTE EITHER DATABASE'S VARIABLES CAN BE ACCESSED

. DISP OFF COST,AMOUNT,ITEM,DATE
      0.75  12.04  Charcoal              800303

. NOTE THE SAME DATABASE CAN BE USED IN BOTH AREAS

. USE SHOPLIST
```

. NOTE BUT ONE MUST BE CAREFUL SINCE THE VARIABLE NAMES ARE IDENTICAL
. NOTE IN BOTH DATABASES

SET

a. SET <parm1> [ON]
[OFF]
b. SET <parm2> TO <opt>

This command changes the configuration of dBASE. SET has two forms. Form a allows those parameters that are "toggles" to be set on or off; form b allows those parameters that need one of the different strings described below to have its default reset.

Form a parameters and defaults:

	<parm1>	action	meaning
1.	ECHO	ON	all commands which come from a command file are echoed on the screen.
		OFF	There is no echo.
2.	STEP	ON	dBASE halts after the completion of each command and waits for the user to decide either to go to the next command, quit (escape) from the command file, or enter a command from the keyboard. (STEP is used for debugging command files).
		OFF	Normal operations are resumed.
3.	TALK	ON	The results from commands are displayed on the screen.
		OFF	There is no display shown.
4.	PRINT	ON	Output is echoed to printer.
		OFF	The echo is turned off.
5.	CONSOLE	ON	Output is echoed to the screen.
		OFF	Output to the screen is turned off.

Note: the default values are underlined.

6.	ALTERNATE	ON	Output is echoed to a disk file.
		OFF	The echo to the file is turned off.

7. SCREEN	ON	Full-screen operations are turned on for APPEND, INSERT, EDIT, and CREATE
	OFF	Full-screen operations are turned off.
8. LINKAGE	ON	Makes all sequential commands (LIST, REPORT, SUM, i. e. commands that have a <scope> parameter) perform positioning on both the PRIMARY and SECONDARY databases.
	OFF	Makes PRIMARY and SECONDARY databases independant.
9. COLON	ON	Bounds GET data items with colons in @ commands.
	OFF	Removes colons.
10. BELL	ON	Bell rings whenever illegal data is entered or data boundaries are crossed.
	OFF	Bell is turned off.
11. ESCAPE	ON	An escape character (1B Hex) aborts execution of command files.
	OFF	There is no escape.
12. EXACT	ON	Requires that character strings match completely (except for trailing blanks) in expressions and the FIND command.
	OFF	Matches will be made on the basis of the length of the second string, e.g. "ABCDEF" = "ABC" is true.

for OFF set bits to 27 41 (with APINSTAL)
for ON set bits to 27 40

	SET	
13. INTENSITY	ON	Full-screen operations will use dual intensity screen characters (normal and inverse video on some terminals)
	OFF	Dual intensity will not be used.
14. DEBUG	ON	Output from the ECHO and STEP commands will be sent to the printer so that full-screen commands may be checked out without the screen becoming cluttered.
	OFF	No extra output on the printer.
15. CARRY	ON	Data from the previous record will be carried-over when APPENDing records in the full-screen mode.
	OFF	No carrying will be done.
16. CONFIRM	ON	dBASE will not skip to next field in full-screen editing until a control key (like return) is typed.
	OFF	dBASE will skip to next field anytime too many characters are entered.
17. EJECT	ON	REPORT command will eject a page before beginning a new report.
	OFF	The page eject will be suppressed.
18. RAW	ON	Places spaces between fields when the DISPLAY and LIST commands are used without the fields list.
	OFF	Spaces are left off.
19. SCREEN	ON	Uses full-screen for EDIT, APPEND, INSERT and CREATE commands.
	OFF	Turns full-screen capabilities off.

Form b parameters and their formats:

1. SET HEADING TO <string>

This form of the SET command saves the <string> internally and prints the string as part of the report header line. The <string> can be up to 60 characters long. (See REPORT for an example.)

2. SET FORMAT TO [SCREEN]
[PRINT]
[<format file>]

The first two forms of this SET parameter determine where the output of "@" commands will go. The last form determines where @ commands are READ from. (See the "@" and READ commands.)

3. SET DEFAULT TO <drive>

This SET commands makes the specified disk drive into the default drive. dBASE will assume that inexplicit file names are on this disk drive. This allows command files to be written in such a way (conveniently) that referenced files may be on any drive in the system. This can also be done with &-macros for further generality in disk drive assignment. In the interactive mode of dBASE, this SET command permits implicit file names.

When a default drive has been set, ALL inexplicit filenames are set to the dBASE default. This includes form files, command files, memory files, format files, index files, text files as well as database files.

The parameter <drive> may or may not have the colon (:) attached, that is, both "B" and "B:" are acceptable forms of specifing which drive is wanted,

NOTE: This SET command does not affect the CP/M default drive in any way. The dBASE initial default drive is the same as the CP/M default drive, the SET DEFAULT redefines dBASE's internal default only while within dBASE.

Example:

```
SET DEFAULT TO B:

USE DATEVSTR      (dBASE will access the 'B' drive for
                      this database)
```

4. SET ALTERNATE TO [<file>]

This form of the SET ALTERNATE command is part of a two step process to write everything that is normally written onto the screen, onto a disk file as well. This includes output that dBASE generates as well as all inputs typed onto the console. This form identifies and opens the receiving disk file. If the <file> existed on the disk prior to this command, it will be overwritten. A subsequent SET ALTERNATE ON begins the echo process.

Example:

```
SET ALTERNATE TO B:PRINTFLE
SET ALTERNATE ON
   .
   .
   .
any commands
   .
   .
   .
SET ALTERNATE TO anyfile
```

Everything which appears on the screen or printer will be copied onto (in this example) B:PRINTFLE.TXT, which can be word processed, printed, or saved.

5. SET DATE TO mm/dd/yy

The system date can be set or reset at any time with this command. It however does not perform date/calendar validation like the date request when dBASE is first started.

```
SET DATE TO 12,10,76
```

6. SET INDEX TO <index file> [, <index file>, ... <index file>]

SET INDEX TO identifies and sets up as many as seven index files to be used for future operations. If an index file is currently in USE when this command is issued then the old index file is closed and the new one established.

Note: when the new index is set up, the database is left positioned where it was, but, the index does not point anywhere. A FIND command or GOTO must be issued to set the index pointer, before any commands that have a next clause are issued.

The first index file named is considered as the Master Index. All FINDs use only this index and the database will be in the Master Index order (when skipping).

A "SET INDEX TO" command (with no index files) will release all indexes and the database will be a sequential file.

7. SET MARGIN TO n

This form of the SET command allows the user to control the left margin when a report is printed. All lines to be printed will be offset by n spaces. The n parameter must be a literal number in the range 1 to 254.

SKIP

SKIP [+][<exp>]
[-]

This command causes the current record pointer to be advanced or backed up relative to its current location.

Example:

```
   USE INVNTRY1

. 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
00010   973328    0     27 CAN COVER         0.73 21     17    5    0.99

. 5

. SKIP -2
RECORD: 00003

. SKIP
RECORD: 00004

. SKIP 3
RECORD: 00007
```

SORT

```
SORT ON <field> TO <file> [ASCENDING ]
                         [DESCENDING]
```

This command allows the user to sort data files to another file which is different from the original file. The file in USE is sorted on one of the data fields and may be sorted into ascending or descending order. Notice that the USE file remains in USE and is unaltered.

While the SORT command allows only one key, a database may be sorted on several keys by cascading sorts: sort on the most minor key first and progress toward the major key. dBASE will only disturb the order of records when necessary. The collating sequence for character fields is the ASCII code. ASCENDING is assumed if neither ASCENDING or DESCENDING is specified.

The sort uses the ASCII collating sequence. This means that the string 'SMITH' is "smaller" than 'Smith' (the expression "'SMITH' < 'Smith'" would be TRUE).

The INDEX command is contrasted with the SORT command in this way: INDEX, when done, performs nearly all of SORTs dutys. Also, INDEX generally allows greater freedom and greater speed than SORT.

```
. USE SHOPLIST

. LIST
00001  BEANS #303 CAN          5       0.75
00002  BREAD LOAVES            2       0.97
00003  T-BONE STEAKS           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 (1/2 GAL)          2       1.30
00009  CHARCOAL, 5# BAGS       2       0.75

. SORT ON ITEM TO SORTFILE
SORT COMPLETE

. USE SORTFILE
```

```
. LIST
00001 BEANS #303 CAN          5    0.75
00002 BLEU CHEESE             1    1.96
00003 BREAD LOAVES            2    0.97
00004 CHARCOAL, 5# BAGS       2    0.75
00005 LETTUCE                 2    0.53
00006 MILK (1/2 GAL)          2    1.30
00007 PAPER PLATES            1    0.86
00008 PLASTIC FORKS           5    0.42
00009 T-BONE STEAKS           4    3.94
```

STORE

STORE <exp> TO <memvar>

This command computes the value of an expression and stores the value into a memory variable. If the memory variable did not exist before this command was issued then dBASE will create the memory variable automatically.

Note that STORE will alter only memory variables. Use the REPLACE command to change database field variables.

```
. RELEASE ALL

. STORE 1 TO ONE
  1

. STORE 'ABCDEFGHIJKL' TO ALFABET
ABCDEFGHIJKL

. STORE ALFABET+' NEW STUFF' TO CHARS
ABCDEFGHIJKL NEW STUFF

  STORE ONE*1.0000 TO ONE
 1.0000

. DISPLAY MEMORY
EOF         (L)  .T.
ONE         (N)   1.0000
ALFABET     (C)  ABCDEFGHIJKL
CHARS       (C)  ABCDEFGHIJKL NEW STUFF
** TOTAL **       04 VARIABLES USED  00042 BYTES USED
```

SUM

SUM <field> [,<field>] [TO <memvar list>]
[<scope>] [FOR <exp>]

The SUM command adds numeric expressions involving the USE file according to the <scope> and FOR clauses. Up to 5 expressions may be simultaneously summed. If the TO clause is present, the sums are also stored into memory variables (memory variables will be created if they didn't exist prior to the issuance of the sum command). The default scope of SUM is all non-deleted records.

```
. USE SHOPLIST

. LIST
00001   BEANS #303 CAN          5       0.75
00002   BREAD LOAVES            2       0.97
00003   T-BONE STEAKS           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 (1/2 GAL)          2       1.30
00009   CHARCOAL, 5# BAGS       2       0.75

. SUM COST
 11.48

. SUM COST FOR NO=1
 2.82

. SUM COST,NO
 11.48  24

. SUM COST TO MSUM
 11.48

. ? MSUM
 11.48

. DISPLAY MEMORY
MSUM        (N)    11.48
** TOTAL **        01 VARIABLES USED  00006 BYTES USED

. ? MSUM*1.10
12.6280

. SUM NO*COST,NO,COST,COST/NO
 31.53  24 11.48 5.81
```

TOTAL

TOTAL ON <key> TO <database> [FIELDS <list>] [FOR <expression>]

The TOTAL command is similar to the subtotal capability in the REPORT command except that the subtotals are placed into a database instead of printed. This allows condensation of data by eliminating detail and summarizing.

Note: the USE database must be either presorted by the key or indexed on the key.

If the TO database was defined (if it existed and had a structure), then it's structure will be left intact and used to decide which fields will be totalled arithmetically.

If the TO database did not exist prior to this TOTAL command, then the structure will be constructed using the field names given by the FIELDS phrase. If there is no FIELD phrase then the structure from the USE database will be copied to the TO file.

This command is most selective when the TO database exists and the FIELD phrase is included in the command. In this case, only the numeric fields in the FIELDS are totalled. In any other configuration of this command, all numeric fields are totalled.

TOTAL can also be used to remove duplicate records from a database since a non-numeric field in the FIELDS list is not totalled (naturally) and is not flagged as an error.

Example:

```
. USE ORDERS INDEX ORDERS

. DISPLAY 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
00003  HARRIS, ARNOLD        11528    44
00007  JUAN, DON             21828     5
00001  SWARTZ, JOE           31415    13
00005  MACK, JAY             31415     3
00008  SALT, CLARA           70296     9
00002  SWARTZ, JOE           76767    13
00006  TERRY, HANS           76767     5
00004  ADAMS, JEAN           89793    12
```

(Imagine that the warehouse needs to know how many of each item to bring out. By totaling on the quantity as long as the part numbers are the same, a database is generated that contains part numbers and the number needed)

(The database CALLS has already been defined)

```
. TOTAL ON PART:NO TO CALLS
00006 RECORDS COPIED

. USE CALLS

. DISP STRU
STRUCTURE FOR FILE:   CALLS.DBF
NUMBER OF RECORDS:    00006
DATE OF LAST UPDATE:  00/00/00
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH  DEC
001      PART:NO      C    005
002      AMOUNT       N    005
** TOTAL **             00011

. LIST
00001  11528    44
00002  21828     5
00003  31415    16      (Note: two orders totaled)
00004  70296     9
00005  76767    18      (Note: two other orders totaled)
00006  89793    12
```

UPDATE

```
UPDATE FROM <database> ON <key> [ADD <field list>]
                        [REPLACE <field list>]
```

The UPDATE command revises the USE file by using data from a second database to modify the USE database. Updated items can be summed or replaced in entirety. A record is updated when the criterion is met by the comparison of a field in the USE database with one from the FROM database. These fields are known as the key and are supplied with the ON phrase.

Note: the USE database must be either pre-sorted by the key or indexed on the key. The FROM database must be pre-sorted by the key.

Both databases are 'rewound' and a record is read. If the keys match, the add or replace action takes place as directed. If the key in the USE file is smaller (in sort sequence) than the key in the FROM database, then no action takes place, and the record is skipped and left unchanged. Similarly, if the FROM key is smaller, no updates happen and that record is skipped.

Example:

```
  USE INVUPDAT

  DISPLAY STRUCTURE
STRUCTURE FOR FILE:  INVUPDAT.DBF
NUMBER OF RECORDS:   00003
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME       TYPE WIDTH   DEC
001      PART:NO     C    005
002      ON:HAND     N    005
003      COST        N    010     002
** TOTAL **              00021

. LIST
00001  21828    77      35.88
00002  70296     0     250.00
00003  89793     2  134999.00
```

(Notice that the database is sorted on the "key" PART:NO.)

```
. USE INVENTRY INDEX INVENTRY

. DISPLAY STRUCTURE
STRUCTURE FOR FILE:  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

. DISP ALL
00008  #9 COAL                    22.00 11528    16
00005  SINK, KITCHEN              34.72 21828    77
00001  TIME STITCH                 9.99 24776     1
00002  WIDGET                      1.67 31415    18
00007  RINGS, GOLDEN             200.00 70296     5
00006  TROMBONES                 198.37 76767    76
00004  TANK, SHERMAN          134999.00 89793     5
00003  GADGET, LARGE              16.33 92653     7
```

(Again notice that the database is indexed on the "key" PART:NO.)

```
. UPDATE ON PART:NO FROM INVUPDAT      ON:HAND REPLACE COST

. LIST
00008  #9 COAL                    22.00 11528
00005  SINK, KITCHEN              35.88 21828   154
00001  TIME STITCH                 9.99 24776     1
00002  WIDGET                      1.67 31415    18
00007  RINGS, GOLDEN             250.00 70296     5
00006  TROMBONES                 198.37 76767    76
00004  TANK, SHERMAN          134999.00 89793     7
00003  GADGET, LARGE              16.33 92653
```

(Note--the two new Sherman tanks were added to the database and the cost of the golden rings and the kitchen sinks were replaced with the new prices.)

USE

USE [<database file>]
USE <databasefile> INDEX <index file> [, <index file>, ... <index file>]

Example:

. USE DATABASE INDEX NAME,CITY,PART:NO,SALESMAN

The USE command specifies which (pre-existing) database file is to be the file in USE. If there was a USE file prior to this command, the old file is closed. If a filename is not specified in the command, then the previous USE file is closed.

The second form of USE is to specify a database for operation and an associated index file (which was previously created by the INDEX command or the SET INDEX TO <index file> command) and permits subsequent index operations such as FIND and indexed sequential file access.

Up to seven index files may be USEd with any one database at the same time. The first index file named is considered as the Master Index. All FINDs use only this index and the database will be in the Master Index order (when skipping). All of the named index files will be automatically updated anytime their keys are modified (by APPEND, EDIT, REPLACE, READ, or BROWSE commands).

Examples:

. USE EXAMPLE

. USE TRACE INDEX TRACE

WAIT

WAIT [TO <memvar>]

This command causes dBASE to cease operations until any character is entered from the keyboard, the message WAITING is displayed on the screen. If the TO clause is specified, then the single keystroke that releases dBASE from the wait-state will be entered into the memory variable.

The TO option is most useful when only a single character is required to direct the action of a command file process e.g. menu selections. Notice that a carriage return is not necessary to "send" the character as in the ACCEPT and INPUT commands.

If any non-printable character (i.e. RETURN, LINE FEED, or any other control character) is typed as the response to a WAIT TO command, the value of the memory variable is set to a blank.

Example:

```
. RELEASE ALL

. WAIT TO ACTION
WAITING 1

. DISP MEMO
ACTION      (N)   1
** TOTAL **       01 VARIABLES USED  00006 BYTES USED
```

APPENDIX A COMMAND FILE EXAMPLE

The following is one example of how command files may be used in a practical environment. In this example, the command files are used like a program written in a more classical language. Command files can contain groups of commands which perform some smaller function e.g. a series of SORT's.

This example is a simple checkbook balancing and check register maintenance system. It consists of 4 command files: the controlling file, MENU, and three subordinate files, NEWENTR, CANCELS, and BALANCE. This problem solution could be structured in many different ways; here, this example has been structured to show the dBASE commands that deal especially with command files.

The command files were created by a text editor using the type ".CMD" in order to facilitate their usage. The sample run is an actual output of dBASE using the SET ALTERNATE technique. Refer to the SET command for this technique.

In solving any database problem, one should first consider what data fields will be required. For this example, the following fields were selected:

NO - the check number
TO - the recipient of the check
AMT - the dollar amount of the check
CAN - the cancelled/not-cancelled status of a check
DATE - the date on which the check was written

dBASE is then entered to CREATE the database structure.

```
. CREATE
FILENAME:CHECKREG
ENTER RECORD STRUCTURE AS FOLLOWS:
 FIELD    NAME,TYPE,WIDTH,DECIMAL PLACES
 001      NO,N,4
 002      TO,C,30
 003      AMT,N,10,2
 004      CAN,L
 005      DATE,C,10
 006      (cr)
INPUT NOW?N
```

A text editor is then executed and the following command file sources are entered:

First the MENU command file;

```
NOTE -  Example dBASE Command file program
*
*
SET TALK off
USE CHECKREG
DO WHILE T
   ?
   ?
   ?
   ? '          Checkbook Balancer Menu'
   ?
   ?
   ? '          0 - EXIT'
   ? '          1 - Enter New Checks'
   ? '          2 - Enter Cancelled Checks'
   ? '          3 - Balance'
   ?
   ? ' enter desired action'
   WAIT TO ACTION
   IF ACTION='0'
      SET TALK on
      CANCEL
   ENDIF
   IF ACTION='1'
      DO NEWENTR
   ENDIF
   IF ACTION='2'
      DO CANCELS
   ENDIF
   IF ACTION='3'
      DO BALANCE
   ENDIF
ENDDO
RETURN
```

Second the NEWENTR command file

```
NOTE - NEWENTR Command File to Enter New Checks
*
REMARK Enter Check Number of 0 to Exit
DO WHILE T
   ?
   ?
   INPUT "Enter Check Number       " to C:NO
   IF C:NO=0
      RETURN
   ENDIF
   ?
   ACCEPT "Paid to Order of        " to C:TO
   INPUT "Amount of Check          " to C:AMT
   ACCEPT "Date of Check            " to C:DAT
   ?
   INPUT "Are all fields correct ? " to GO:NOGO
   IF .NOT.GO:NOGO
      LOOP
   ENDIF
   APPEND BLANK
   REPLACE NO with C:NO, TO with C:TO, AMT with C:AMT, DATE ;
     with C:DAT, CAN with F
ENDDO
```

Third the CANCELS command file

```
NOTE - CANCELS Command file to enter cancelled checks
*
REMARK  Enter Check Number of 0 to Exit
DO WHILE T
   ?
   INPUT "Enter Cancelled Check no " to C:CAN
   IF C:CAN=0
      RETURN
   ENDIF
   GO TOP
   LOCATE for C:CAN=NO
   REPLACE CAN with T
ENDDO
```

Last the BALANCE command file

```
NOTE - BALANCE Command File to Balance Checkbook.
*
SUM AMT to OUTSTAND for .NOT.CAN
?
?
DISPLAY off 'Total Outstanding Checks = $',OUTSTAND
?
REMARK  Enter Outstanding Deposits, Enter 0 to Proceed
STORE T to ACTIVE
STORE 1 to COUNT
STORE 0 to T:OUT
DO WHILE ACTIVE
   STORE STR(COUNT,3) to I
   INPUT 'Enter Amount of Outstanding Deposit &I ' to D:OUT
   IF D:OUT=0
      STORE F to ACTIVE
   ELSE
      STORE D:OUT+T:OUT to T:OUT
      STORE COUNT+1 to COUNT
   ENDIF
ENDDO
DISPLAY OFF COUNT-1,' Total Outstanding Deposits   Total = $',T:OUT
?
INPUT "Enter Ending Balance" to BEGIN
DISPLAY OFF 'Current Balance = $',BEGIN+T:OUT-OUTSTAND
WAIT
RETURN
```

A sample run of these command files follows:

```
. DO MENU


          Checkbook Balancer Menu


          0 - EXIT
          1 - Enter New Checks
          2 - Enter Cancelled Checks
          3 - Balance

 enter desired action
WAITING 1
 Enter Check Number of 0 to Exit


Enter Check Number       :1000

Paid to Order of         :ACME Rentals
Amount of Check          :123.45
Date of Check            :10 Jun 79

Are all fields correct ? :y


Enter Check Number       :1001

Paid to Order of         :Mag Publishing Co.
Amount of Check          :79.88
Date of Check            :12 Jun 79

Are all fields correct ? :y


Enter Check Number       :1002

Paid to Order of         :Radon Inert Gases
Amount of Check          :86.86
Date of Check            :13 Jun 79

Are all fields correct ? :y
```

```
Enter Check Number       :1003

Paid to Order of         :Neuron Comm. Inc.
Amount of Check          :723.31
Date of Check            :14 Jun 79

Are all fields correct ? :y


Enter Check Number       :1004

Paid to Order of         :Crankshaft Auto
Amount of Check          :2753.47
Date of Check            :19 Jun 79

Are all fields correct ? :y


Enter Check Number       :0


         Checkbook Balancer Menu


         0 - EXIT
         1 - Enter New Checks
         2 - Enter Cancelled Checks
         3 - Balance

 enter desired action
WAITING 2
  Enter Check Number of 0 to Exit

Enter Cancelled Check no :1001

Enter Cancelled Check no :1003

Enter Cancelled Check no :0


         Checkbook Balancer Menu


         0 - EXIT
         1 - Enter New Checks
         2 - Enter Cancelled Checks
         3 - Balance

 enter desired action
WAITING 3
Total Outstanding Checks = $ 2983.78

  Enter Outstanding Deposits, Enter 0 to Proceed
```

```
Enter Amount of Outstanding Deposit   1 :1234.56
Enter Amount of Outstanding Deposit   2 :.03
Enter Amount of Outstanding Deposit   3 :333.44
Enter Amount of Outstanding Deposit   4 : 0
  3 Total Outstanding Deposits   Total = $ 1568.03

Enter Ending Balance:1445.89
Current Balance = $ 50.14
WAITING


         Checkbook Balancer Menu


         0 - EXIT
         1 - Enter New Checks
         2 - Enter Cancelled Checks
         3 - Balance

 enter desired action
WAITING 0
DO CANCELLED
```

At this point, the user could easily do direct dBASE commands to interrogate, modify, or report on the database file. For instance the commands:

```
 DISPLAY DATE,AMOUNT for NO=1003
```

or

```
 SUM AMT for DATE>'01 Jun'
```

or any other dBASE commands could be issued to provide information as needed to accommodate unforeseen circumstances in the course of managing a checkbook.

```
? <exp> [,<exp>]
@ <coordinates> [SAY <exp> [USING '<picture>']] [GET
        <variable> [PICTURE '<picture>']]
ACCEPT ["<cstring>"] TO <memvar>
APPEND [FROM <file> [SDF] [DELIMITED] [FOR <exp>]]
        or [BLANK]
BROWSE
CANCEL
CHANGE FIELD <list> [<scope>] [FOR <exp>]
CLEAR [GETS]
CONTINUE
COPY TO <file> [<scope>] [FIELD <list>] [FOR <exp>]
        [SDF] [DELIMITED [WITH <delimiter>]] or [STRUCTURE]
COUNT [<scope>] [FOR <exp>] [TO <memvar>]
CREATE [<filename>]
DELETE [<scope>] [FOR <exp>]
DELETE FILE <file>
DISPLAY [<scope>] [FOR <exp>] [<exp list>] [OFF]
DISPLAY STRUCTURE
DISPLAY MEMORY
DISPLAY FILES [ON <disk drive>] [LIKE <skeleton>]
DO <file>
DO WHILE <exp>
EDIT
EJECT
ELSE
ENDDO
ENDIF
ERASE
FIND <key>
GO or GOTO [RECORD], or [TOP], or [BOTTOM], <n>
IF <exp>
INDEX ON <char string expression> TO <index file name>
INPUT ["<cstring>"] TO <memvar>
INSERT [BEFORE], or [BLANK]
JOIN TO <file> FOR <expression> [FIELDS <field list>]
LIST
LOCATE [<scope>] [FOR <exp>]
LOOP
MODIFY STRUCTURE
MODIFY COMMAND <command file>
NOTE or *
PACK
QUIT [TO <list of CP/M level commands or .COM files>]
READ
RECALL [<scope>] [FOR <exp>]

RELEASE [<memvar list>], or [ALL]
REMARK
RENAME <current file name> TO <new file name>
REPLACE [<scope>] <field> WITH <exp> [AND <field> WITH <exp>]
REPORT [<scope>] [FORM <form file>] [TO PRINT] [FOR <exp>]
RESET
```

```
RESTORE
RETURN
SAVE TO <file>
SELECT [PRIMARY or SECONDARY]
SET <parm> [ON], or [OFF]
SET ALTERNATE TO <file>
SET DEFAULT TO <drive>
SET DATE TO <string>
SET FORMAT TO <format file name>
SET HEADING TO <string>
SET INDEX TO <index file>
SET MARGIN TO <n>
SKIP <+/-> [<n>]
SORT ON <field> TO <file> [ASCENDING], or [DESCENDING]
STORE <exp> TO <memvar>
SUM <field> [<scope>] [TO <memvar list>] [FOR <exp>]
TOTAL TO <file> ON <key variable> [FIELDS <field list>]
UPDATE FROM <file> ON <key variable> [ADD <field list>]
     [REPLACE <field list>]
USE <file> [INDEX <index file name>]
WAIT [TO <memvar>]
```

FUNCTIONS:

@(<string1>,<string2>)	AT function
*	deleted record func
#	record number func
!(<char string>)	upper case function
$(<char string>,<start>,<length>)	substring function
<string1>$<string2>	substring search
CHR(<numeric expression>)	numeric to ASCII
DATE()	system date func
EOF	end-of-file func
FILE(<file>)	existance func
INT(<numeric expression>)	integer function
LEN(<char string>)	length function
STR(<numeric expression>,<width>[,<decimals>])	string func
VAL(<char string>)	value function
TRIM(<char string>)	trims strings
TYPE(<exp>)	supplies data type

APPENDIX C LIMITATIONS AND CONSTRAINTS

number of fields per record	32 max
number of characters per record	1000 max
number of records per database	65535 max
number of characters per character string	254 max
accuracy of numeric fields	10 digits
largest number	1.8 x 10**63 approx
smallest number	1.0 x 10**-63 approx
number of memory variables	64 max
number of characters per command line	254 max
number of expressions in SUM command	5 max
number of characters in REPORT header	254 max
number of characters in index key	100 max
number of pending GETS	64 max
number of files open at one time	16 max

APPENDIX D ERROR MESSAGES

BAD DECIMAL WIDTH FIELD

BAD FILE NAME
Syntax error in filename.

BAD NAME FIELD

BAD TYPE FIELD
Must be C, N, or L.

BAD WIDTH FIELD

CANNOT INSERT - THERE ARE NO RECORDS IN DATABASE FILE
Use the APPEND command instead.

CANNOT OPEN FILE
Internal error, contact dealer for support.

COMMAND FILE CANNOT BE FOUND
Check spelling.

DATA ITEM NOT FOUND

DATABASE IN USE IS NOT INDEXED
FIND is only permitted on indexed databases.

DIRECTORY IS FULL
The CP/M disk directory cannot hold anymore files.

DISK IS FULL

END OF FILE FOUND UNEXPECTEDLY
The database in USE is not in the correct format. If all records are correct and present, then PACK and re-INDEX the database.

"FIELD" PHRASE NOT FOUND

FILE ALREADY EXISTS

FILE DOES NOT EXIST

FILE IS CURRENTLY OPEN
Type a USE or CLEAR command to close the file.

FORMAT FILE CANNOT BE OPENED

FORMAT FILE HAS NOT BEEN SET

ILLEGAL DATA TYPE

ILLEGAL GOTO VALUE

ILLEGAL VARIABLE NAME
Only alphanumerics and colons are allowed in variable and field names.

INDEX DOES NOT MATCH DATABASE
dBASE cannot match the key with the database. Try another index file.

INDEX FILE CANNOT BE OPENED
Check spelling or INDEX the database.

JOIN ATTEMPTED TO GENERATE MORE THAN 65,534 RECORDS
The FOR clause allows too many joined output records, make it more stringent.

KEYS ARE NOT THE SAME LENGTH

MACRO IS NOT A CHARACTER STRING
&macros must be character strings.

MORE THAN 5 FIELDS TO SUM

NESTING LIMIT VIOLATION EXCEEDED

NO EXPRESSION TO SUM

NO "FOR" PHRASE

NO "FROM" PHRASE

NO FIND
More a diagnostic type message than an error message. dBASE couldn't find the key.

NON-NUMERIC EXPRESSION

NONEXISTENT FILE

"ON" PHRASE NOT FOUND

OUT OF MEMORY FOR MEMORY VARIABLES
Reduce the number or size of memory variables.

RECORD LENGTH EXCEEDS MAXIMUM SIZE (OF 1000)

RECORD NOT IN INDEX
Index file was not updated after a record was added. Reindex.

RECORD OUT OF RANGE
Record number greater than number of records in database. The Record doesn't exist.

SORTER INTERNAL ERROR, NOTIFY SCDP
Internal error, contact dealer for support.

SOURCE AND DESTINATION DATA TYPES ARE DIFFERENT

*** SYNTAX ERROR ***

SYNTAX ERROR IN FORMAT SPECIFICATION

SYNTAX ERROR, RE-ENTER

"TO" PHRASE NOT FOUND

TOO MANY CHARACTERS

TOO MANY FILES ARE OPEN
There is a maximum of 16 files allowed to be open at one time.

TOO MANY MEMORY VARIABLES
There is a maximum of 64 memory variables

TOO MANY RETURNS ENCOUNTERED
Probably an error in the structure of a command file.

"WITH" PHRASE NOT FOUND

UNASSIGNED FILE NUMBER
Internal error, contact dealer for support.

*** UNKNOWN COMMAND

VARIABLE CANNOT BE FOUND
Need to create the variable, or check the spelling.

INDEX

NOTES

Additional user data about dBASE II operation not yet included in the Manual.

1. The 0th line on the screen is now reserved for special purposes. Therefore, do not issue a format command like '@ 0,<y> SAY <exp>'

2. The REPORT command has a limit of 24 data fields.

3. Under MP/M the QUIT TO <filename> will not operate.

4. PACK will not reduce amount of disk space reserved for that file by CP/M. To recover the space, use a COPY TO <filename> and then delete the source file. This is a limitation of the CP/M operating system not of dBASE II.

5. DO NOT RENAME a file in USE. Generally it is not even a good practice to RENAME a file while under command program control.

6. The proper syntax for the COPY STRUCTURE command is:
   ```
   USE <file>
   COPY STRUCTURE TO <newfile>
   ```
   the 'STRUCTURE' option should immediately follow the verb 'COPY'.

7. When calling a dBASE data file into USE, do not use the '.DBF' extension. dBASE adds this extension automatically.