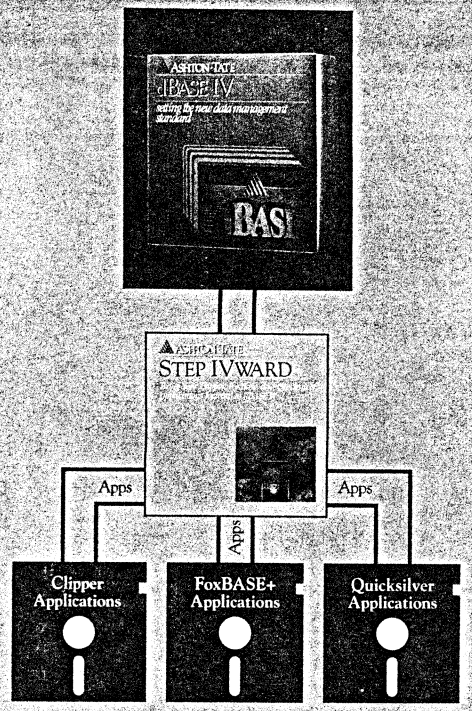
IVWARD™

)ASE+ or Quicksilver applications up to
performance offered by dBASE IV™ with
l, from Tate Publishing, a division of
vity users and application developers the
convert applications written in dBASE®
ge of the new features and functionality of
nanagement standard.
mand at the DOS prompt or by using the
tem, Step IVward will automatically
ASE+ or Quicksilver applications to run
V users now have a whole new world of
ns to choose from. And application
xisting and future applications available to
er community.
SE applications written in dBASE-like
advantages that dBASE IV has to offer.

d options include:

s – Step IVward automatically creates new
ating date and time file was converted to

tep IVward will automatically process all
es and UDFs of an application. This
of large multi-file applications.
IVward leaves original code as comments
y conversion reference.
p IVward places comments in converted
for conversion.
the code of your converted dBASE IV
tructure checking.



l Computer AT, Personal System/2 or

ents:
rsion 3.1 or higher or 100% compatibles

nimum capacity floppy disk drive

Display Requirements:
- Monochrome or color graphics (CGA, EGA or VGA)

Printer: (optional)
- Any printer with at least 80 columns

Software Requirements:
- Application source code written in Clipper, FoxBASE+ or Quicksilver
- dBASE IV required to run converted applications (640KB system required to run dBASE IV.)

$89.95

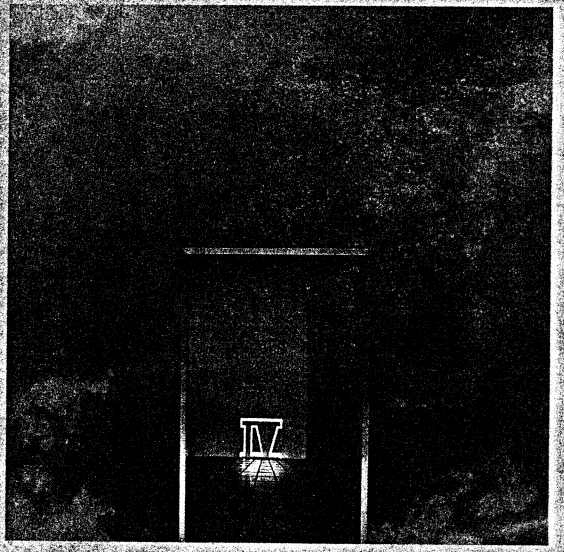
ep IVward

ASHTON-TATE

ASHTON-TATE

STEP IVWARD

Bring Your Clipper, FoxBase, and Quicksilver applications forward to dBASE IV



STEP IVWARD

Bring Your Clipper, FoxBASE+, And Quicksilver Applications Forward To dBASE IV

ASHTON·TATE®

Step IVward

Acknowledgement

Ashton-Tate would like to thank Buzzwords International, Inc., for their assistance in the development of Step IVward. Special thanks are extended to Rich Comeau, President, and to Don Keller, for their commitment.

10/88

Contents

Chapter 1: Getting Started

System and Source-Program Requirements

To run Step IVward, you need these system requirements:

- A DOS operating system version 3.1 or greater *or* a NETBIOS-based operating system
- 640K of memory (more is recommended)
- At least two disk drives (a hard disk is recommended)
- An IBM PC-type monochrome, color, EGA, or VGA screen display

An 80-column printer is useful but not required, since Step IVward messages may be sent to the screen and/or disk. These messages are very important for after-the-fact program modifications.

Source program requirements are:

- Presence of Step IVward and a Clipper, FoxBASE + , or a Quicksilver source code program.
- Presence of any source file in ASCII format for Step IVward. Step IVward will not work on compiled versions of your program.

WARNING
Before you begin using Step IVward, be sure to:
- Make a back up of your source code.
- Make two working copies of your Step IVward disk, which is not copy protected. Put the original copy away for safekeeping, and use a working copy.
- Examine the file Readme.1st. Step IVward is constantly being improved, and addendum to this manual appears in this file.

DOS Installation

Under the DOS environment, create a subdirectory called **STEP**. If you plan to use Step IVward frequently, modify your Autoexec.bat file to include a path to the STEP subdirectory.

Assuming your hard disk is drive C, follow these steps:

1. Make a directory called STEP.

   C > md step

2. Change to the STEP directory.

   C > cd \step

3. Copy all of the Step IVward files on drive A to drive C.

   C > copy a:*.* c:

NOTE
If you have only one application, you will get faster system operation by copying the complete Step IVward disk into your application subdirectory. It takes DOS much less time to scan the subdirectory that you are logged into than to search the path subdirectories for Step IVward.

Running Step IVward

You can run Step IVward in one of two ways: from the Step IVward menu system, or as a single-line batch process.

Step IVward menus

To use the Step IVward menu system, type step at the DOS prompt. Then use the main menu to select the various options available in the program.

Typing step causes the main menu system to appear, allowing you to change option settings and evaluate your program interactively. The menus allow you to control all aspects of program operation. They are described in Chapter 2.

To analyze all .prg files in a directory, use the asterisk (*) wildcard option in the **Input File Name** option of the **Setup** menu.

Step IVward batch process

To execute Step IVward as a batch process, type step followed by the filename, the language parameter, then any of the optional parameters.

The filename and language parameter are required. The filename must include an extension and may include the asterisk (*) wildcard option. The language parameter indicates the language of the program to translate. The valid options are: /c for Clipper, /f for FoxBASE +, or /q for Quicksilver.

For example, from the DOS prompt, type step sample1.prg /c causes a Clipper file named Sample1.prg on the logged drive to be translated. To convert a FoxBASE + program, type step sample2.prg /f. To convert a Quicksilver program, type step sample3.prg /q.

All optional parameters are always preceded with a slash (/). Batch parameters correspond to the menus in Step IVward. The first character of a parameter is the first character of the menu name. The second part of the parameter matches the option offset on the pull-down menu. Chapter 2 describes each menu option in detail. The valid batch parameters are summarized in Table 1-1.

When a batch parameter is included, Step IVward switches the option from its default setting to its complementary setting. If you have a previously saved configuration, it is ignored when running Step IVward in batch processing mode.

Table 1-1 Batch parameters

Parameter	Menu
/c	Clipper
/f	FoxBASE +
/q	Quicksilver
/S2	Output filename
/S3	Programmer's name
/S4	Copyright notice
/O1	Suppress file headers
/O2	Suppress file footer
/O3	Process files tree
/O4	Prompt before each file
/O5	Comment old source code
/P1	Printer
/T1	Suppress display during translation

Results file

Step IVward always summarizes the error and warning messages to a file named Warning.txt. Warning.txt is a report of program names, the line numbers on which error or warning messages occurred, and messages generated.

A Sample Clipper Translation

This section shows an example of a Clipper routine and the program generated by Step IVward. The program file, Example.prg, was translated in a batch process. The command line includes the program filename, the language parameter, and three additional parameter to suppress the translation display, write an output file, and include a copyright notice.

```
C:\STEP>step example.prg /c /t1 /s2 /s4 Buzzwords, Intl., Inc., 1988
```

Original code

```
* Example.prg
*
SET KEY 28 TO USERHELP
USE Names INDEX Zipcode
mrecords = LASTREC()
PUBLIC avalues[mrecords], acnt[mrecords]
GO TOP
row = 1
DO WHILE .NOT. EOF()
   cnt = 0
   STORE Zipcode TO avalues[row]
   DO WHILE Zipcode = avalues[row] .AND. .NOT. EOF()
      cnt = cnt + 1
      SKIP
   ENDDO
   STORE cnt TO acnt[row]
   row = row + 1
END
WAIT "Display array values? (y/n)" TO minput
maction = IF(UPPER(minput) = "Y","D",IF(UPPER(minput) = "N","X","Z"))
IF maction = "D"
   FOR x = 1 TO row - 1
         ? avalues[x], acnt[x]
   NEXT
ELSEIF maction = "X"
   ? "There were " + ALLTRIM(STR(row - 1) + " unique codes."
ELSE
? "Leaving"
END
CLEAR ALL
RETURN
* eop: Example.prg
```


Translated code

```
* Converted to dBASE IV by Step IVward 1.0 11/12/88 7:22 PM
* FILENAME: EXAMPLE.PRG
* BY:
* NOTICE: Buzzwards, Intl., Inc., 1988
* Example.prg
*
ON KEY LABEL F1 DO USERHELP
USE Names INDEX Zipcode
mrecords = RECCOUNT()
PUBLIC ARRAY avalues[mrecords], acnt[mrecords]
GO TOP
STORE 1 TO row
DO WHILE .NOT. EOF()
   cnt = 0
   STORE Zipcode TO avalues[row]
   DO WHILE Zipcode = avalues[row] .AND. .NOT. EOF()
      cnt = cnt + 1
      SKIP
   ENDDO
   STORE cnt TO acnt[row]
   STORE row+ 1 TO row
ENDDO
WAIT "Display array values? (y/n)" TO minput
maction = IIF(UPPER(minput) = "Y","D",IIF(UPPER(minput) = "N","X","Z"))
DO CASE
CASE maction = "D"
   x = 1
   DO WHILE x <= row- 1
      ? avalues[x], acnt[x]
      x = x + 1
   ENDDO
CASE maction = "X"
   ? "There were " + LTRIM(RTRIM((row- 1) + " unique codes."))
OTHERWISE
   ? "Leaving"
ENDCASE
CLEAR ALL
RETURN
* eop: Example.prg
* Converted to dBASE IV by Step IVward 1.0 11/12/88 7:22 PM
```

Chapter 2:
Step IVward Menus

The five Step IVward menus, **Setup**, **Options**, **Print**, **Translate**, and **Exit**, are shown in Figure 2-1. For a complete description of the menus and options, see the appropriate section that follows.

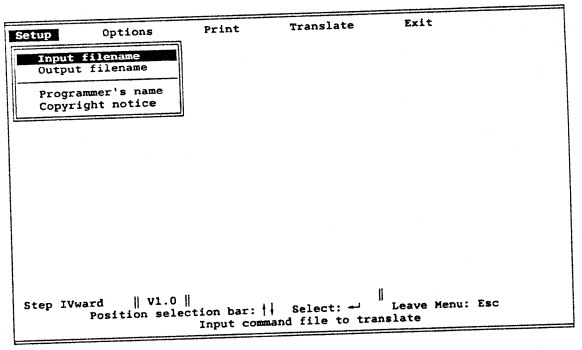


Figure 2-1 The Step IVward menus

Use ← and → to navigate to the menus. Use the ↓ and ↑ to move the highlight to a menu option. Press ↵ on an option to see and modify the default setting. To finish changing an option, press ↵.

The Setup Menu

The **Setup** menu options (Figure 2-2) let you enter the name of the input file to be translated, the output file after it is translated and indented, the programmer's name, and the copyright notice. When you start the Step IVward program, the **Setup** menu opens.

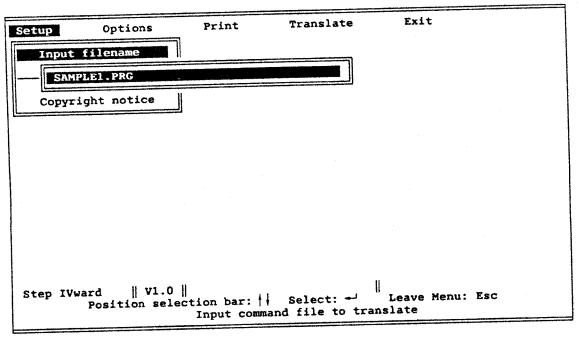


Figure 2-2 The Setup menu

Input filename

This option allows you to select the source code file (.prg) to be translated. If Step IVward is not in your application directory, specify the directory. To translate all .prg files in a directory, use the * wildcard option (*.prg). See also "Process files tree" under "The Options Menu" in this chapter for more on files specification.

When Step IVward is processing files trees, only those files referenced with DO commands are translated. If you have a file of UDFs that is not referenced with a DO command, you will need to use link list processing. To process a link-list file, enter the link-list filename with an extension of .lnk. (Be sure that your link list filename also has an extension of .lnk.) Link list processing is recommended since this will provide Step IVward with the names of all files included in your application.

Output filename (/S2)

This option allows you to specify the destination for the tabbed (indented) dBASE IV version of the program file which Step IVward will translate.

The default extension is .prg. For example, if you translate Sample.prg, the default name for the translated file will be Sample.prg. The original .prg is renamed Sample.org if the file is in the same subdirectory.

NOTE
If there is no output filename specified, the source file is not updated. Instead, the output goes to the screen or printer.

Programmer's name (/S3)

This option allows you to automatically add a comment line containing your name at the beginning of all reports. This helps you with hard-copy documentation of your programs. A total of 40 spaces is available.

Copyright notice

This option allows you to add a comment line with your copyright information.

The Options Menu

The choices on the **Options** menu (Figure 2-3) let Step IVward construct the processing functions.

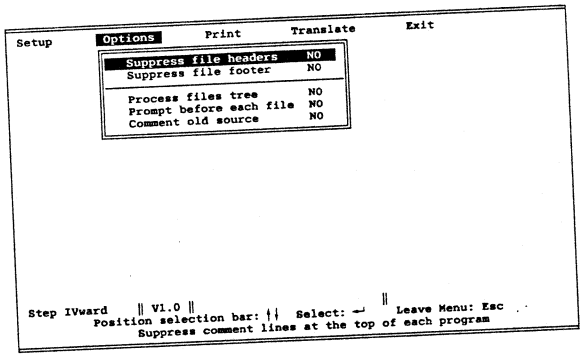


Figure 2-3 The Options menu

Suppress file headers (/O1)

Step IVward inserts a set of comments at the beginning of your new dBASE IV source code. To suppress the heading, set this option ON.

Here is an example:

```
. Converted to dBASE IV by Step IVward V1.0 9-27-1988 0:23 AM
* FILENAME: SAMPLE1.PRG
* BY: RICH COMEAU
* NOTICE: COPYRIGHT 1988 BUZZWORDS INTERNATIONAL INC
* sample1 - Demonstrates Clipper conversion
```

Suppress file footer (/O2)

By default, Step IVward inserts a footer comment at the end of your new dBASE IV source code. To suppress the footer, set this option ON.

Here is an example of a footer:

```
. Converted to dBASE IV by Step IVward V1.0 9-27-1988 0:23 AM
```

Process files tree (/O3)

This option causes Step IVward to process all source code modules that are called by your main start-up program. The default is OFF. See also "Input filename" described under "The Setup Menu" in this chapter for information on link-list processing.

The method of program processing is done by the convention DO Filename shown in this example:

```
.. Main Module
   DO Submenu1
** SUBMENU1 (First Level Down Program File)
   DO Submenu2
** SUBMENU1 (Second Level Down Program File)
   DO Submenu3
```

NOTE
Step IVward does not support macros as filenames when processing the files tree. This is because the expansion of filename *occurs only at run time. To ensure that Step IVward processes all of your files, add the following IF structure to your source code:*

```
DO &Filename.
IF .F.       && Never Executed
   DO File1 && Possible File For Macro Substitution
   DO File2 && Possible File For Macro Substitution
   DO File3 && Possible File For Macro Substitution
ENDIF
```


This syntax forces Step IVward to process all possible program files. Since the IF condition is false, this code will not execute when your program is run. It is recommended that modules that call other modules use explicit filenames. An example is DO Filename. Also, you can ensure that all .prg files in the directory are translated by specifying *.prg as the input filename.

Prompt before each file (/O4)

This option prompts before each file is processed to find out if you want to translate that file or skip it. You will be asked **Translate this file or Skip over it? (T or S)**. The default is OFF.

Comment old source (/O5)

This option inserts your old source code as comments. This default is OFF. By inserting your old code as comments you will find debugging easier. dBASE IV will give you the same operation if your code is inserted as comments due to the compiler. However, the disk space required for new source code with old converted source code as comments is greatly increased.

The Print Menu

The **Print** menu (Figure 2-4) allows you print control of your translation.

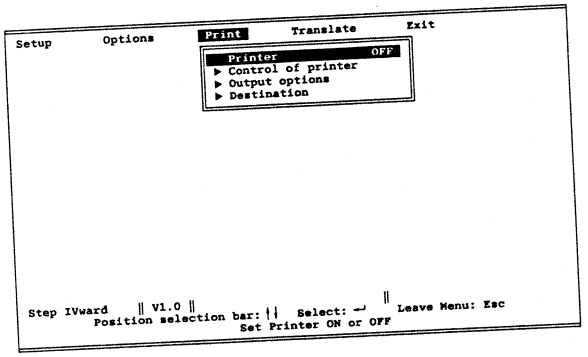


Figure 2-4 The Print menu

Printer (/P1)

This option sets the printer output ON or OFF. The default is OFF, so output goes to the screen.

Control of printer

This option displays the submenu shown in Figure 2-5.

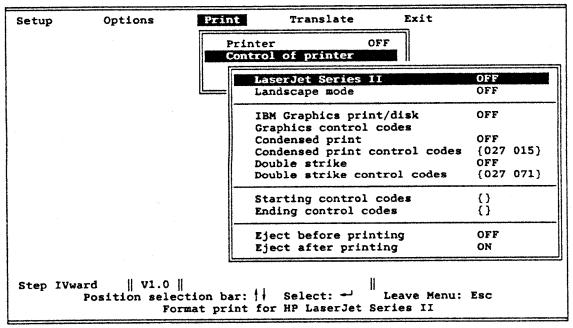


Figure 2-5 The Control of Printer submenu

LaserJet Series II

This option gives you proper output including graphic line character output if your printer is Hewlett-Packard LaserJet Series II compatible. The default is OFF. The line drawing characters are determined by the **Graphics control codes** menu unless **IBM Graphics printer** is ON; then the single bar lines are printed. Setting this option ON automatically sets **IBM Graphics printer** ON.

Landscape mode

This option sets output for landscape mode. It is for LaserJet Series II compatible printers only. The default is OFF.

IBM Graphics printer

This option indicates if the printer is compatible with the IBM graphics set (the graphics character set consists of characters above ASCII 128). This option allows your printer to produce sharp, double bar lines indicating the loops in program logic, exactly as you see them on your IBM display. The default is OFF.

If **IBM Graphics printer** is OFF, the line drawing characters are determined by the **Graphics control codes** settings.

Graphics control codes

The **Graphics control codes** submenu (Figure 2-6) defines the characters used for graphic reports. This is needed for printers that are not IBM Graphics compatible to use graphics characters. The default values can be changed by entering new values in decimal ASCII format.

The seven definable printer characteristic defaults are:

Characteristic	Non-IBM Graphic	IBM Graphic
Enter graphics mode	none	none
Exit graphics mode	none	none
Horizontal line	045 (-)	196 (—)
Vertical line	033 (!)	179 (|)
Upper left corner	046 (.)	218 (┌)
Left side tee	033 (!)	195 (├)
Lower left corner	039 (')	192 (└)

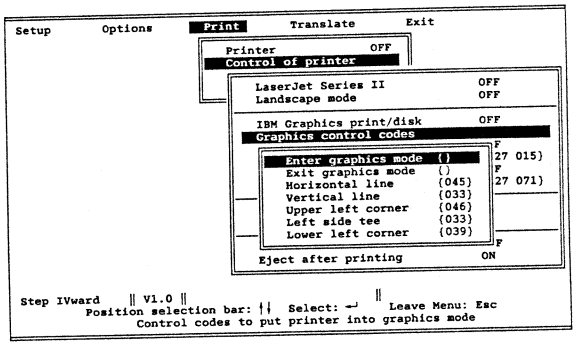


Figure 2-6 The Graphics sequences submenu

Condensed print

This option toggles condensed characters ON and OFF. Step IVward sends the code specified under the **Condensed print control codes** option to your printer. The default is OFF.

Condensed print control codes

This option configures Step IVward for the proper control sequence for condensed print mode. You can enter up to three sets of control codes. The default value is CHR(27) CHR(15).

Double strike control codes

This option toggles double strike characters ON and OFF. Step IVward sends the code specified under the **Double strike control codes** option to your printer. The default is OFF.

Double strike control codes

This option enters the control codes for your printer for double-strike mode. Keep in mind that, if your printer is not IBM Graphics compatible, you must have the decimal codes available from your printer manual to input the values. The default codes are CHR(27) CHR(71).

Starting control codes

This option sets a starting control code sequence for your printer. The default setting is none.

Ending control codes

This option sets an ending control code sequence for your printer. The default setting is none.

Eject before printing

This option causes a page eject before any printing begins. The default is OFF.

Eject after printing

This option causes a page eject after printing the translation. The default is OFF.

Output options

This option displays the submenu shown in Figure 2-7.

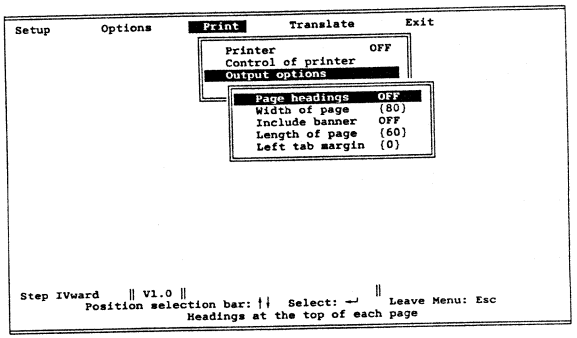


Figure 2-7 The Output options submenu

Page headings

This option puts page headings at the top of each page. This is very useful when you bind your program listings in a book. The filename, page number, date, and time, are printed at the top of each page. The default is OFF.

Width of page

This option sets the width of the printer output to a maximum of a 255 characters. The default is 80.

Include banner

This option produces a banner heading at the top of each listing. The characters are made up of asterisks and are 2 inches tall. The default is OFF.

Length of page

This option sets the number of lines printed on each page when using page headings. The default is 60.

Left tab margin

This option sets the left tab margin. This is very handy when you want to bind your listings. Laser printer users may need to set the output to 132-column output to avoid wrap-around printing. The default is 0.

Destination

This option displays the submenu shown in Figure 2-8.

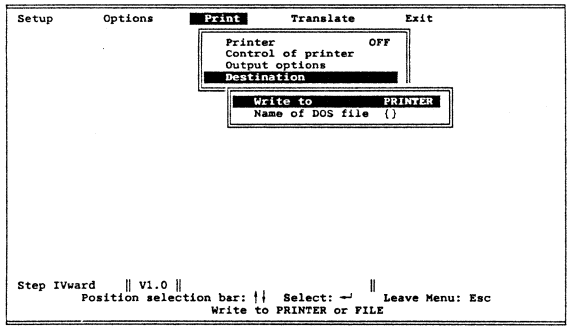


Figure 2-8 The Destination submenu

Write to

This option toggles the print destination between PRINTER and FILE. The default is PRINTER.

Name of DOS file

This option names the print destination file. This selection does not affect the result file or the output filename set under **Setup**.

The Translate Menu

The **Translate** menu (Figure 2-9) allows you control of your translation.

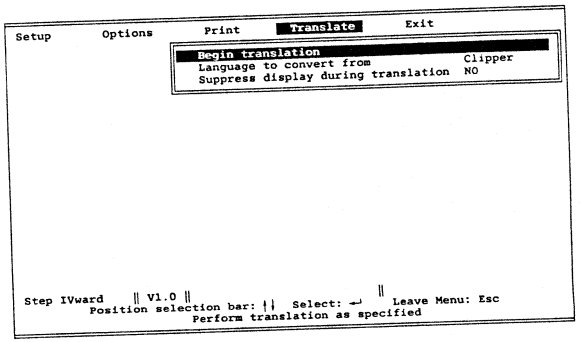


Figure 2-9 The Translate menu

Begin translation

This option starts the translation of your source file. You must have specified an input and output filename on the **Setup** menu before executing translation. You must also specify the language to translate.

Select language to convert from

Clipper (/C)

This option turns on Clipper as the input source language.

FoxBASE + (/F)

This option turns on FoxBASE + as the input source language.

Quicksilver (/Q)

This option turns on Quicksilver as the input source language.

Suppress display during translation (/T1)

This option displays the translated source code to the screen. If you want a faster translation, leave this selection OFF. Screen displays slow down translation but aid in understanding the new source code.

The Exit Menu

The **Exit** menu (Figure 2-10) allows you to exit the translation.

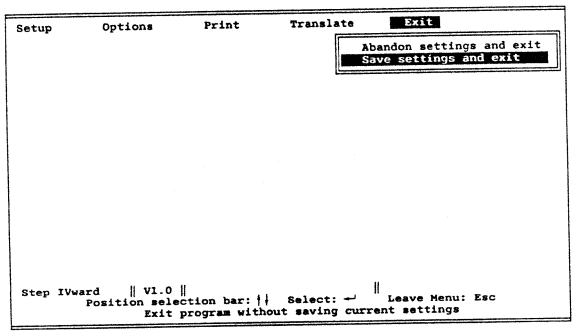


Figure 2-10 The Exit menu

Abandon settings and exit

This option exits you to the operating system. The system configuration is not saved.

Save settings and exit

This option saves your current switch settings to the file Step.cfg and exits to the operating system.

Chapter 3: Clipper to dBASE IV

Clipper Translation Instructions

Translation from Clipper to dBASE IV is a very straight-forward process. These are the steps needed for a proper translation of a complete application:

1. Select **Input filename** on the **Setup** menu by pressing ↵.
2. Type the filename of your main application.
3. Select **Output filename** and press ↵.
4. Type the filename of your main application or link file (the default is the same as your input filename).
5. Select **Process files tree** on the **Options** menu.
6. Select **Select language to convert from** on the **Translate** menu.
7. Select **Clipper** as your input source language.
8. Now select **Begin translation.**

Your code is now translated directly to dBASE IV source code. There will be insertions of LOAD and CALL statements for the included .bin files. Additional user-defined functions (UDFs) are included for the conversion or replacement of several of the Clipper functions. Code lines that cannot be converted directly to either dBASE IV or a binary file are inserted as comments with an appropriate warning or error message.

Essentials

This section covers some essential differences between Clipper and dBASE IV. These are concepts and differences that Step IVward is unable to reconcile.

Program Flow Errors

If a function that cannot be translated is included in the < condition > of a program flow command, Step IVward generates an error message and either comments the line out or, removes the < condition > . The action taken is as follows:

- **The condition is in a DO WHILE or IF statement:**
- **The condition is in a CASE statement:** the condition is removed.

Linked C Functions

Step IVward cannot tell the difference between UDFs and C functions linked into your program. Any C functions must be redone as a binary file.

Command Verbs as Memory Variable Names

Clipper allows a line to begin with a memory variable assignment even if the memory variable name is the same as a command verb. Command lines beginning with memory variables whose names are the same as command lines are converted to STORE statements.

Functions Invoked Without Assignment

Clipper allows functions to be invoked and take some action without returning a value. For example, using INKEY(0) to pause program execution until a key is pressed,

Clipper command syntax

INKEY(0)

dBASE IV command syntax

s4_temp = INKEY(0)

Step IVward converts the Clipper statement to a valid dBASE IV command by inserting making the statement an assignment. The return value is assigned to a dummy memvar name unlikely to be used for anything else.

Relational and Logical Operators

Step IVward converts the following Clipper relational operators.

Description	Clipper syntax	dBASE IV syntax
Not equal to	< =	< >
Exactly equal to	= =	=
Logical not	!	.NOT.
Modulus	%	MOD()

Step IVward issues a warning message when " = = " is encountered in a command line. dBASE IV numeric values uses BCD format for accuracy. The dBASE IV float number format is provided for dBASE III PLUS (and Clipper) compatibility. The dBASE IV numeric type eliminates the need for the = = operator.

Switch Commands

Clipper switch commands optionally support a logical argument. (Switch commands are those SET commands with ON/OFF arguments.) Step IVward converts SET commands with logical arguments to IF structures. For example:

```
SET NEAR <expl>
```

is converted to:

```
IF <expl>
   SET NEAR ON
ELSE
   SET NEAR OFF
ENDIF
```

Printing

Clipper allows some commands to print to both printer and disk file at the same time, dBASE IV does not. If both the TO PRINT and TO FILE options are present, Step IVward converts the following commands into two command lines:

DISPLAY

LIST

LABEL FORM

REPORT FORM

TYPE

Parameter Passing

Clipper allows programs to receive parameters passed form DOS via a PARAMETERS statement, dBASE IV does not. If a PARAMETERS statement is detected in the beginning of the root module, Step IVward comments the command line out.

dBASE IV requires a predetermined number of parameters be passed to a PARAMETER's statement. Clipper allows a variable number of parameters to be passed.

Miscellaneous Differences

Here are some differences between Clipper and dBASE IV that you should be aware of:

- To ensure Clipper compatibility, Step IVward adds the commands SET TALK OFF, SET SAFETY OFF, and SET STATUS OFF to the root module.
- Clipper allows nested macros, dBASE IV does not. Step IVward does not detect this condition.
- dBASE IV has a limit of 10 work area, Clipper supports 250. See the SELECT command for more information.
- Arrays in Clipper can have up to 4096 elements. dBASE IV support up to 1024 elements per array.
- dBASE IV database files can have up to 255 fields per .dbf file, Clipper supports 1028 fields.
- Clipper allows database specific commands do be executed with no database file in use. Also, @...SAY coordinates may be off the screen dimensions in Clipper. Step IVward does not detect these conditions and dBASE IV will generate an error at run time.
- In Clipper, **CTRL-U** performs an *undo* of changes in the current @...GET. In dBASE IV, **CTRL-U** flags the current record for deletion.
- Character fields in Clipper can have up to 64k of data. dBASE IV character fields have a maximum length of 254.
- Clipper and dBASE IV index files may have same extension, .ndx. Index files must be rebuilt in dBASE IV.

Clipper Commands

This section lists Clipper commands that do not have analogs in dBASE IV and describes how Step IVward converts these commands.

@ ... BOX

@...BOX draws a box on the screen and fills it with a specified character.

Syntax

Clipper syntax

@ <expN1>, <expN2>, <expN3>, <expN4> BOX (<expC>)

dBASE IV syntax

@ <expN1>, <expN2> TO <expN3>, <expN4> <expC>

Result

In both languages, <expC> represents a border definition string, but the formats are different. In Clipper the first eight characters define the box borders starting from the upper-left corner and then proceed clockwise (tl,t,tr,r,br,b,bl,l). The ninth character is a fill character. In dBASE IV, the order of the first eight is (t,b,l,r,tl,tr,bl,br). The values must be separated by commas. There is no fill character is dBASE IV. Step IVward drops the border definition string. This results in a single-line border.

If the border expression is a variable name, the variable is dropped and a warning message is generated. To use the variable, you must include a macro symbol (&) and drop the fill character parameter from the definition string.

Example

Clipper code

```
frame = CHR(201)+CHR(205)+CHR(187)+CHR(186)+CHR(188)+CHR(205)+CHR(200)+CHR(186)+CHR(176)
@ 1,0,24,79 BOX frame
```

dBASE IV code

```
frame = "CHR(205),CHR(205),CHR(186),CHR(186),CHR(201),CHR(187),CHR(200),CHR(188)"
@ 1,0 TO 24,79 &frame.
```

@ ... PROMPT

@...PROMPT defines and displays menu prompts to be activated by MENU TO.

Syntax

Clipper

@ <expN1>, <expN2> PROMPT <expC> [MESSAGE <expC>]

dBASE IV

DEFINE MENU <menu name>
DEFINE PAD <pad name> OF <menu name> PROMPT <expC>
[AT <row>, <col>] [MESSAGE <expC>]

or

DEFINE POPUP <pop-up name> FROM <row1>, <col1>
[TO <row2>, <col2>]
DEFINE BAR <expN> OF <pop-up name> PROMPT <expC>
MESSAGE <expC>

Result

Step IVward converts this command to either a pop-up or menu system, depending on the <col> values. If all of the <col> values are the same, Step IVward translates the menu to a pop-up menu system; otherwise, a menu bar system is created. The <col> values are evaluated starting with the first @...PROMPT command encountered until the first MENU TO command. The arguments are converted as follows:

Menu bar system

<row>, <col>
DEFINE PAD...AT coordinates
<expC1>
DEFINE PAD <pad name>
<expC2>
DEFINE PAD...MESSAGE <expC>

Pop-up menu system

<row>, <col>
The top left corner of the pop-up
<expC1>
DEFINE BAR...PROMPT <expC>
<expC2>
DEFINE BAR...MESSAGE <expC>

BEGIN SEQUENCE

This command structure typically defines user-defined error scoping within a program file.

Result

dBASE IV has no corresponding command structure. Step IVward comments out the BEGIN SEQUENCE and END command lines and changes BREAK to RETURN TO MASTER.

CLEAR

CLEAR erases the screen, homes the cursor, and releases all pending GETs.

Syntax

Clipper

CLEAR [SCREEN]

dBASE IV

CLEAR

Result

In dBASE IV, CLEAR is equivalent to CLEAR SCREEN. Step IVward removes the SCREEN option if it is present. If CLEAR only is detected, Step IVward adds the additional command CLEAR GETS.

COMMIT

COMMIT flushes all Clipper buffers and performs a solid-disk write for all work areas.

Syntax

Clipper

COMMIT

dBASE IV

dBASE IV has no corresponding command. The SET AUTOSAVE command in dBASE IV saves each record to disk after I/O has been performed on the record.

Result

Step IVward converts COMMIT to the following dBASE IV routine:

```
SET AUTOSAVE ON
SKIP 0
SET AUTOSAVE OFF
```

CREATE

Creates an empty structure extended database file.

Syntax

Clipper

CREATE <filename>

dBASE IV

dBASE IV does not create an empty structure extended database file. Instead, the dBASE IV CREATE command gives you access to the database file design.

Result

Step IVward replaces CREATE with a procedure that creates an empty structure extended file.

END

Clipper permits ENDIF, ENDDO, and ENDCASE to be abbreviated to END.

Syntax

Clipper syntax

```
IF              DO WHILE        DO CASE
<commands>      <commands>      <commands>
END             END             END
```

dBASE IV syntax

```
IF              DO WHILE        DO CASE
<commands>      <commands>      <commands>
ENDIF           ENDDO           ENDCASE
```

Result

Step IVward converts the keyword END to ENDIF, ENDDO, or ENDCASE as appropriate.

EXTERNAL

EXTERNAL declares symbol(s) for the linker.

Result

Step IVward comments the command line out and generates an error message. Also, the procedure list is added to the list of modules to process if **Files tree processing** is selected.

FIND

Searches for the first record in an indexed database file with a key that matches a specified expression.

Syntax

Clipper

FIND <literal> /(<expC>)

dBASE IV

```
FIND <literal>
SEEK <exp>
```

Result

All FIND commands are converted to SEEK commands.

FOR ... NEXT

Syntax

Clipper

```
FOR <memvar> = <expN1> TO <expN2> [STEP <expN3>]
    <commands>
   [EXIT]
    <commands>
NEXT
```

dBASE IV

```
<memvar> = <expN1>
DO WHILE <memvar> <= <expN2>
   <commands>
  [EXIT]
   <commands>
   <memvar> = <memvar> + <expN3>
ENDDO
```

Results

If the STEP argument is negative, the DO WHILE test is changed to >=. If <expN3> evaluates to a negative value, Step IVward will not be able to detect this condition. You will have to change the operators manually.

FUNCTION

FUNCTION declares a user-defined function.

Syntax

Clipper

FUNCTION <procedure>

dBASE IV

FUNCTION <procedure>
PROCEDURE <procedure>

Result

dBASE IV supports user-defined functions (UDFs) by beginning the UDF with the FUNCTION command verb. However, due to the differences in Clipper and dBASE IV UDFs, Step IVward translates some Clipper UDFs to dBASE IV PROCEDURES. (For information on dBASE IV UDFs, see the dBASE IV *Language Reference* manual.) The rules for translation are as follows:

Step IVward checks to see if any dBASE IV restricted commands are included in the UDF. The structure is left unchanged. If restricted commands are present, the RETURN expression is removed and a previously declared variable is given the return value. Then, all statements encountered that contain this function are preceded with a call to the procedure. In the offending command line, function is replaced with the memory variable.

IF

A structured programming command that enables conditional processing of commands.

Syntax

Clipper

IF <condition1>
  <commands>
ELSEIF <condition2>
  <commands>
ELSE
  <commands>
ENDIF

dBASE IV

DO CASE
CASE <condition1>
  <commands>
CASE <condition2>
  <commands>
OTHERWISE
  <commands>
ENDCASE

Result

Step IVward converts Clipper IF...ELSEIF constructs to DO CASE structures.

KEYBOARD

KEYBOARD stuffs the keyboard buffer with a character string.

Syntax

Clipper

KEYBOARD <expC>

dBASE IV

dBASE IV does not support the KEYBOARD command.

Result

KEYBOARD is replaced by a call to a binary routine. In dBASE IV you can record a keystroke macro that, when replayed with the PLAY MACRO command, will stuff all recorded keys into the keyboard buffer.

MENU TO

Executes a menu for the currently defined set of PROMPTS.

Syntax

Clipper

MENU TO <memory variable>

dBASE IV

ACTIVATE MENU <menu name>
ACTIVATE POPUP <popup name>

Result

Step IVward converts MENU TO to the appropriate dBASE IV menu activation command. The new command is dependent on the type of menu system Step IVward built for the currently defined prompts.

PRIVATE

PRIVATE allows you to create local memory variables in a lower-level program with the same names as memory variables that were created in a calling program or were previously declared PUBLIC.

Syntax

Clipper

PRIVATE <memvar list>

dBASE IV

PRIVATE <memvar list>

Result

Unlike Clipper, dBASE IV does not allow arrays to specifically be declared PRIVATE. If an array is detected in the command line, the array name is left in the statement but, the brackets are removed. A DECLARE statement is then added to initialize the array.

PUBLIC

Designates specified memory variables and arrays that you can use and alter in any dBASE program or subprogram.

Syntax

Clipper

PUBLIC <memvar list> [,clipper]

dBASE IV

PUBLIC <memvar list> /ARRAY <array element list>

Result

dBASE IV does not allow memory variables and arrays to be declared PUBLIC in the same command. Step IVward declares arrays PUBLIC in a subsequent PUBLIC statement. If clipper is detected in a command line, Step IVward sets clipper to true immediately after the PUBLIC command.

SAVE

SAVE stores all or part of the current set of memory variables and array elements to a disk file.

Syntax

Clipper

SAVE TO <filename> [ALL [LIKE/EXCEPT <skeleton>]]

dBASE IV

SAVE TO <filename> [ALL LIKE/EXCEPT <skeleton>]

Result

dBASE IV saves all memory variables unless the ALL LIKE or EXCEPT options are included. If Step IVward encounters ALL without the LIKE or EXCEPT options, ALL is removed from the command line.

SELECT

SELECT chooses a work area in which to open a database file, or to specify a work area in which a database file is already open.

Syntax

Clipper

SELECT < work area name/alias >

dBASE IV

SELECT < work area name/alias >

Result

dBASE IV has a limit of 10 work areas. If Step IVward detects a work area reference greater than 10, the statement is commented out and an error message is generated. If SELECT 0 is detected, Step IVward changes the statement to SELECT SELECT().

SET COLOR/COLOUR TO

Define the color for the next screen display output.

Syntax

Clipper syntax

SET COLOR/COLOUR TO (< expC >)

dBASE IV syntax

SET COLOR TO < color list >

Result

dBASE IV requires that < color > list be either a literal or macro expansion; an indirect reference is not allowed. If an indirect reference is used, Step IVward attempts to convert it to a macro substitution. If COLOUR is detected, it is changed to COLOR. All other Clipper SET COLOR TO arguments are supported in dBASE IV.

If numbers are included to denote the color settings, Step IVward converts them to the appropriate letter code for dBASE IV. If you are using macro expansion, you will have to change the memory variable declaration to remove the numbers.

SET CURSOR

This command toggles the screen cursor on and off.

Syntax

Clipper

SET CURSOR ON/off/(< expL >)

dBASE IV

dBASE IV has no corresponding command. dBASE IV automatically turns the cursor on or off, depending on the command executing.

Result

Step IVward converts SET CURSOR to a CALL to a binary file. See the "Essentials" section at the beginning of this chapter for more information on switch commands.

SET KEY

Allows a procedure to be executed from any wait state.

Syntax

Clipper

SET KEY < expN > TO [< procedure >]

dBASE IV

ON KEY [LABEL < key label name >] [< command >]

Result

Clipper SET KEY TO allows a procedure to be executed from any wait state when a designated key is pressed. In Clipper, the return key value is designated by a number. Step IVward converts the numbers to the dBASE IV key label names.

Commands called by ON KEY have the same restrictions as dBASE IV UDFs.

SET MESSAGE

This command sets the line for Clipper menu messages.

Result

dBASE IV has no corresponding command. dBASE IV menu messages always appear on line 24.

Step IVward comments the command line out and generates a warning message.

SET PRINTER TO

Syntax

Clipper

SET PRINTER TO [< DOS device > / < filename > /(< expC >)]

dBASE IV

SET PRINTER TO [< DOS device >]
SET PRINTER TO [FILE < filename > /(< expC >)]

Result

If the output is direct to a file, Step IVward adds the keyword FILE. dBASE IV does not allow printer output to be directed to a DOS device and a disk file at the same time. If both are specified, Step IVward comments the line out and an error message is generated.

SET RELATION TO

Relates database files in multiple work areas using a key expression or record number.

Syntax

Clipper

SET RELATION [ADDITIVE] TO [< exp >] INTO < alias > [, < exp >]
INTO < alias >]]...

dBASE IV

SET RELATION TO [< exp >] INTO < alias > [, < exp >] INTO
< alias >]]...

Result

dBASE IV does not support the Clipper ADDITIVE option. In dBASE IV, all relations must be specified in the same statement. Step IVward removes ADDITIVE and generates a warning message.

SET SOFTSEEK

This command positions the database file to the record immediately following the potential location of the sought-after key in the file.

Syntax

Clipper

SET SOFTSEEK on/OFF/(< expL >)

dBASE IV

SET NEAR on/OFF

Result

Step IVward converts the keyword SOFTSEEK to NEAR when ON or OFF is specified. See the "Essentials" section at the beginning of this chapter for more information on switch commands.

SET WRAP

This command toggles cursor wrapping in menus.

Result

dBASE IV has no corresponding command. dBASE IV always allows cursor wrapping in menus.

Step IVward comments the command line out and generates a warning message. See the "Essentials" section at the beginning of this chapter for more information on switch commands.

TEXT

TEXT outputs a block of text to the screen, printer, or text file.

Syntax

Clipper

TEXT [TO PRINT][TO FILE <filename>]
<text characters>
ENDTEXT

dBASE IV

TEXT
<text characters>
ENDTEXT

Result

dBASE IV does not support the TO PRINT or TO FILE option of the Clipper TEXT command. Step IVward converts the TO FILE by bounding the TEXT...ENDTEXT construct with SET ALTERNATE TO and CLOSE ALTERNATE commands to redirect output to a text file. If TO PRINT is included, the command structure is bounded by SET PRINT ON and SET PRINT OFF.

Unlike Clipper, dBASE IV does not allow expansion of macros within the TEXT construct. Step IVward does not detect the & within your text and no error is generated.

TOTAL

TOTAL sums the numeric fields of the active database file and creates a second database file to hold the results.

Syntax

Clipper

TOTAL ON <key expr> TO <filename> [FIELDS <field list>]

dBASE IV

TOTAL ON <key field> TO <filename> [FIELDS <field list>]

Result

dBASE IV does not allow totals on key expressions. If an operation is detected in the key expression, the command line is commented out and an error message is generated.

TYPE

TYPE displays the contents of an ASCII text file.

Syntax

Clipper

TYPE <filename> [TO PRINTER][TO FILE <filename>]

dBASE IV

TYPE <filename> [TO PRINTER/TO FILE <filename>]

Result

dBASE IV does not allow the output of TYPE to be sent to the printer and disk file at the same time. If both arguments are included in the original command line, Step IVward converts the command to two separate command lines, one for each direction.

UPDATE

UPDATE uses data from another database file to replace fields in the current database file. It makes changes by matching records in the two files based on a related key.

Syntax

Clipper

UPDATE ON <key exp> FROM <alias> REPLACE <field1> WITH
<exp1> [,<field2> WITH <exp2>]...[RANDOM]

dBASE IV

UPDATE ON <key field> FROM <alias> REPLACE <field1> WITH
<exp1> [,<field2> WITH <exp2>]...[RANDOM]

Result

dBASE IV does not allow updates on key expressions. If an operation is detected in the key expression, the command line is commented out and an error message is generated.

USE

USE opens an existing database file and optionally associated index files.

Syntax

Clipper

USE [< filename >] [INDEX < index file list >]

dBASE IV

USE [< filename >] [INDEX < index file list >]

Result

Clipper allows 15 .ndx files to be open simultaneously in the same work area; dBASE IV allows 10. If more than 10 .ndx files are specified, Step IVward truncates the command line and generates an error message.

NOTE
dBASE IV allows you to have up to 10 .ndx and .mdx files open per work area. Each .mdx file can contain up to 47 index tags. (Index tags are similar to .ndx files.) If your application requires that more that 10 indexes be open per work area, copy them to tags with the COPY INDEX command.

Clipper Functions

This section lists Clipper functions that do not have analogs in dBASE IV and describes how Step IVward converts them.

ACHOICE()

Executes a pop-up menu using an array of character string choices.

Clipper syntax

ACHOICE(< expN1 > , < expN2 > , < expN3 > , < expN4 > , < array1 > [, < array2 > [, < expC > [, < expN5 > [, < expN6 >]]]])

Result

ACHOICE() is converted to the dBASE IV popup-menu defining commands DEFINE POPUP and DEFINE BAR. Only the first four arguments are converted. The arguments are translated as follows:

< expN1 >	< row1 > of DEFINE POPUP
< expN2 >	< col1 > of DEFINE POPUP
< expN3 >	< row2 > of DEFINE POPUP
< expN4 >	< col2 > of DEFINE POPUP
< array1 >	The array name becomes the popup menu name.
< array[< n >] >	< n > is the DEFINE BAR number, the value is the PROMPT < expC >

ACOPY()

Copies elements from one array to another.

Clipper syntax

ACOPY(< array1 > , < array2 > [, < expN1 > [, < expN2 > [, < expN3 >]]])

Result

ACOPY() is converted to a DO WHILE loop that copies all of the elements of the source array to the destination array.

ADEL()

ADEL() deletes an array element.

Clipper syntax

ADEL(< array1 > , < expN1 >)

Result

Converted to DO WHILE loop of array[ndx] = array[ndx + 1]. The value of the last element is then set to a logical false.

ADIR()

Fills a series of arrays with directory information and/or returns the number of files matching a skeleton

Clipper syntax

ADIR(< directory skeleton > [< array1 > [, < array2 > [, < array3 > [, < array4 > [, < array5 >]]]]])

Result

ADIR() is converted to a DO WHILE loop with CALLs to a binary routine.

AFIELDS()

Fills a series of arrays with field names, types, lengths, and decimals.

Clipper syntax

AFIELD([< array1 > [, < array2 > [, < array3 > [, < array4 >]]]])

Result

Converted to LIST STRUCTURE TO FILE Stepwrk.wrk. Then the FOPEN() function and a DO WHILE loop with the FREAD() function get the required information.

AFILL()

Fills an array with a specified value.

Clipper syntax

AFILL(< array > , < exp > , [< expN1 > [, < expN2 >]])

Result

AFILL() is converted to DO WHILE loop of array[ndx] = value.

AINS()

Inserts an undefined element into an array.

Clipper syntax

AINS(< array > , < expN >)

Result

Converted to DO WHILE loop where array[ndx] = array[ndx-1]. Array[ndx1] is set to a logical false (.F.).

ALLTRIM()

Removes leading and trailing blanks.

Clipper syntax

ALLTRIM(< expC >)

Result

ALLTRIM() is converted to LTRIM(RTRIM(< expC >)).

ALTD()

Controls the Clipper debugger.

Clipper syntax

ALTD(< expN >)

Result

dBASE IV does not support this function. Step IVward comments out command lines that contain this function. The dBASE IV Debugger allows multiple break points to be set while debugging a program.

ASCAN()

Scans an array for a value.

Clipper syntax

ASCAN(< array > , < exp > [, < expN1 > [, < expN2 >]])

Result

ASCAN() is converted to DO WHILE loop with comparisons to each element in the array.

ASORT()

Sorts an array or a portion of an array in ascending order.

Clipper syntax

ASORT(< array > , [, < expN1 > [, < expN2 >]])

Result

ASORT() is converted to a user-defined function.

BIN2I()

Converts a character string formatted as a 16-bit signed integer to a Clipper numeric value.

Result

BIN2I() is not supported in dBASE IV. Statements containing BIN2I() are commented out and a warning message is generated by Step IVward.

BIN2L()

Converts a character string formatted as a 32-bit signed integer to a Clipper numeric value.

Result

BIN2L() is not supported in dBASE IV. Statements containing BIN2L() are commented out and a warning message is generated by Step IVward.

BIN2W()

Converts a character string formatted as a 16-bit unsigned integer to a Clipper numeric value.

Result

BIN2W() is not supported in dBASE IV. Statements containing BIN2W() are commented out and a warning message is generated by Step IVward.

CURDIR()

Determines the current DOS directory on specified drive.

Clipper syntax

CURDIR([< expC >])

Result

CURDIR() is converted to a call to a binary routine.

DBEDIT()

Displays and edits records from one or more work areas in a browse-style layout within a defined window area.

Clipper syntax

DBEDIT([< expN1 >][, < expN2 >][, < expN3 >][, < expN4 >]
[, < array1 >][, < expC >][, < array2 >][, < array3 >][, < array4 >]
[, < array5 >][, < array6 >][, < array7 >])

Result

DBEDIT() is converted to a BROWSE command. The first four arguments are used to define a window for the BROWSE. The first if the first array is present, the values of the array are used to build a SET FIELDS field list. The rest of the arguments are ignored by Step IVward.

In dBASE IV, BROWSE uses the active format file to determine data entry formats and validity.

DBFILTER()

Determines the expression of the active filter.

Result

dBASE IV does not support DBFILTER(). Commands that include DBFILTER() are commented out and a warning message is generated by Step IVward.

DBRELATION()

Determines the linking expression of a specified relation.

Result

dBASE IV does not support DBRELATION(). Commands that include DBRELATION() are commented out and a warning message is generated by Step IVward.

DBRSELECT()

Determines the target work area of a specified relation.

Result

dBASE IV does not support DBRSELECT(). Commands that include DBRSELECT() are commented out and a warning message is generated by Step IVward.

DESCEND()

Creates and SEEKs descending order indexes.

Clipper syntax

DESCEND(< exp >)

Result

DESCEND() is converted to a dBASE IV user-defined function.

DOSERROR()

DOSERROR() returns last DOS error number.

Result

dBASE IV does not support DOSERROR(). Commands that include DOSERROR() are commented out and a warning message is generated by Step IVward.

EMPTY()

Determines if an expression evaluates to an empty value.

Clipper Syntax

EMPTY()

Result

EMPTY() is converted to a dBASE IV user-defined function with a CASE statement that compares the TYPE() of the input. EMPTY() returns true for the following conditions:

Data type	Test
Character	All spaces
Numeric	Value of zero
Date	The string conversion is all spaces (" / / ")
Logical	Value of false
Memo	MLINES() = 0

ERRORLEVEL()

Returns current DOS error level setting and optionally sets it to a new value.

Result

dBASE IV does not support ERRORLEVEL(). Commands that include ERRORLEVEL() are commented out and a warning message is generated by Step IVward.

FCLOSE()

Closes a file opened with FOPEN() or FCREATE().

Clipper syntax

FCLOSE(< expN >)

Result

FCLOSE() is converted to a call to a binary routine.

FCOUNT()

Returns the number of fields in the current database file structure.

Clipper syntax

FCOUNT()

Result

FCOUNT() is converted to a user-defined function that has a DO WHILE loop to count the number of fields.

FCREATE()

Creates a new file or truncates an existing file to zero length.

Clipper syntax

FCREATE(< expC > [, < expN >]))

Result

FCREATE() is converted to a user-defined function that makes call to a binary routine.

FERROR()

Returns the DOS error from the last file operation or 0 if there was no error.

Clipper syntax

FERROR()

Result

FERROR() is converted to a user-defined function.

FIELDNAME()

Returns the name of a specified field in the current database file.

Clipper syntax

FIELDNAME(< expN >)

Result

FIELDNAME() is the same as the dBASE IV FIELD() function.
FIELDNAME() is converted to FIELD().

FILE()

FILE() verifies the existence of a specified filename on the disk.

Clipper syntax

FILE(< expC >)

Result

Clipper allows wildcards to be included in the expression; dBASE IV does not. If wildcards are included, Step IVward converts FILE() to a call to a binary routine.

FOPEN()

Performs a DOS file open returns the file handle.

Clipper syntax

FOPEN(< expC > [, < expN >])

Result

FOPEN() is converted to a user-defined function that makes a call to a binary routine.

FREAD()

Reads a file opened by FOPEN() or FCREATE().

Clipper syntax

FREAD(< expN1 > , @ < mvarC > , < expN2 >)

Result

FREAD() is converted to a user-defined function that makes a call to a binary routine.

FREADSTR()

Reads a file opened by FOPEN() or FCREATE().

Clipper syntax

FREADSTR(< expN1 > , < expN2 >)

Result

FREAD() is converted to a user-defined function that makes call to a binary routine.

FSEEK()

Sets the DOS file pointer to a new position in a file opened by FOPEN() or FCREATE().

Clipper syntax

FSEEK(< expN1 > , < expN2 > [, < expN3 >])

Result

FSEEK() is converted to a user-defined function that makes a call to a binary routine.

FWRITE()

Writes the contents of a character memory variable to a file opened by FOPEN() or FCREATE()

Clipper syntax

FWRITE(< expN1 > , < mvarC > [, < expN2 >])

Result

FWRITE() is converted to a user-defined function that makes call to a binary routine.

GETE()

Retrieves contents of a DOS environmental variable.

Clipper syntax

GETE(< expC >)

Result

No conversion is required since dBASE IV will recognize this as the short form of GETENV().

HARDCR()

Replaces all soft carriage returns (CHR(141)) with hard carriage returns (CHR(13)) in order to display memo fields containing soft carriage returns.

Clipper syntax

HARDCR(< expC >)

Result

dBASE IV does not support HARDCR(). Commands that include HARDCR() are commented out and a warning message is generated by Step IVward. dBASE IV wraps text automatically based on the SET MEMOWIDTH value.

HEADER()

Determines the length of the header area of the current database file.

Clipper syntax

HEADER()

Result

HEADER() is replaced by a user-defined function which calculates the length as 32 * < number of fields > + 35.

I2BIN()

Converts an integer numeric value to a character string formatted as a binary integer with the least significant byte first.

Result

dBASE IV does not support I2BIN(). Commands that include I2BIN() are commented out and a warning message is generated by Step IVward.

IF()

Returns the result of one of two expression depending on the evaluation of a condition.

Clipper syntax

IF(< expL > , < exp1 > , < exp2 >)

Result

IF() is the same as the dBASE IV IIF() function. IF() is converted to IIF().

INDEXEXT()

Determines if an index file is dBASE compatible or Clipper compatible.

Clipper syntax

INDEXEXT()

Result

INDEXEXT() is converted to a user-defined function. Note, dBASE IV does not utilize Clipper index files.

INDEXKEY()

Returns the key expression of specified index.

Clipper syntax

INDEXKEY(< expN >)

Result

INDEXKEY() is converted to the dBASE IV KEY() function.

INDEXORD()

Determines the position of the controlling index.

Clipper syntax

INDEXORD()

Result

INDEXORD() is converted to a user-defined function. The UDF uses the dBASE IV functions TAG() and ORDER() to find the controlling index.

ISCOLOUR()

Determines if the computer has a color graphics card installed.

Clipper syntax

ISCOLOUR()

Result

ISCOLOUR() is the same as the dBASE IV ISCOLOR() function.
ISCOLOUR() is converted to ISCOLOR().

ISPRINTER()

ISPRINTER() returns logical true if the printer is ready to accept output.

Clipper syntax

ISPRINTER()

Result

ISPRINTER() is the same as the dBASE IV PRINTSTATUS() function.
ISPRINTER() is converted to PRINTSTATUS().

L2BIN()

Converts an integer value to a character string formatted as a 32-bit binary integer.

Result

dBASE IV does not support L2BIN(). Commands that include L2BIN() are commented out and a warning message is generated by Step IVward.

LASTREC()

LASTREC() returns the number of records in the currently selected database file.

Clipper syntax

LASTREC()

Result

LASTREC() is the same as the dBASE IV RECCOUNT() function. LASTREC() is converted to RECCOUNT().

LEN()

Returns length of character expression or the number of elements in a DECLAREd array.

Clipper syntax

LEN(< expC > / < array name >)

Result

dBASE IV LEN() does not support array element count. To determine the length of an array, build a counting loop whose condition is TYPE("array[n]") < > "U".

MEMOEDIT()

Displays or edits character strings and memo fields.

Result

dBASE IV does not support MEMOEDIT(). Command lines that include MEMOEDIT() are commented out and a warning message is generated by Step IVward .

MEMOLINE()

Extracts a formatted line of text from a character string or memo field.

Result

dBASE IV does not support MEMOLINE(). Step IVward converts MEMOLINE() to a call to a procedure.

MEMOREAD()

Reads the contents of a text file from disk as a character string up to 64K bytes long.

Result

dBASE IV does not support MEMOREAD(). Command lines that include MEMOREAD() are commented out and a warning message is generated by Step IVward .

MEMOTRAN()

Replaces carriage return/line feed pairs in a character string or memo field with a specified character.

Clipper syntax

MEMOTRAN(< expC >)

Result

dBASE IV does not support MEMOTRAN(). Step IVward comments the line out and generates an error message.

MEMOWRIT()

Writes a character string to a specified disk file.

Clipper syntax

MEMOWRIT(< expC1 > , < expC2 >)

Result

dBASE IV does not support MEMOWRIT(). Step IVward converts MEMOLINE() to a call to a procedure.

MLCOUNT()

Counts the number of word-wrapped lines in a character string or memo field.

Clipper syntax

MLCOUNT(< expN >)

Result

dBASE IV does not support MLCOUNT(). Step IVward converts MEMOLINE() to a call to a procedure.

MLPOS()

Determines the position of a specified line number in a character string or memo field.

Clipper syntax

MLPOS(< expN >)

Result

dBASE IV does not support MLPOS(). Step IVward converts MEMOLINE() to a call to a procedure.

NETERR()

Returns logical true if a USE, USE...EXCLUSIVE, or APPEND has failed in a network environment.

Clipper syntax

NETERR()

Result

dBASE IV does not support NETERR(). Command lines that include NETERR() are commented out and a warning message is generated by Step IVward. dBASE IV generates an error when a file or record lock cannot be obtained. These errors can be trapped with the ON ERROR command and the ERROR() function.

NETNAME()

Returns the workstation id as a 15-byte character string.

Clipper syntax

NETNAME()

Result

NETNAME() is converted to a user-defined function that makes call to a binary routine.

NEXTKEY()

Reads the next keystroke without removing it from the keyboard buffer.

Result

dBASE IV does not support NEXTKEY(). Command lines that include NEXTKEY() are removed.

PCOUNT()

Determines the number of actual parameters passed to a procedure or UDF.

Clipper syntax

PCOUNT()

Result

PCOUNT() is converted to the number of parameters on the PARAMETERS statement.

PROCLINE()

Returns the current source code line number.

Clipper syntax

PROCLINE()

Result

PROCLINE() is the same as the dBASE IV LINENO() function. PROCLINE() is converted to LINENO().

PROCNAME()

Returns the name of the current program or procedure.

Clipper syntax

PROCNAME()

Result

PROCNAME() is the same as the dBASE IV PROGRAM() function. PROCNAME() is converted to PROGRAM().

RAT()

Searches a character string for the last instance of a specified substring and returns the starting position as a numeric value.

Clipper syntax

RAT(< expC1 > , < expC2 >)

Result

RAT() is converted to a user-defined function which makes repeated calls to the AT() function.

READEXIT()

Toggles the ↑ and ↓ keys as READ exit keys.

Clipper syntax

READEXIT([< expL >])

Result

dBASE IV does not support READEXIT(). Command lines that include READEXIT() are commented out and a warning message is generated by Step IVward. In dBASE IV, ↑ and ↓ always exit a READ.

READINSERT()

Reports the current insert mode setting for READ and MEMOEDIT() and optionally toggles it *on* or *off*.

Clipper syntax

READINSERT([< expL >])

Result

dBASE IV does not support READINSERT(). Command lines that include READINSERT() are commented out and a warning message is generated by Step IVward.

READVAR()

Determines the current GET/MENU variable for context-sensitive help.

Clipper syntax

READVAR()

Result

READVAR() is the same as the dBASE IV function VARREAD(). READVAR() is converted to VARREAD().

RESTSCREEN()

Redisplays a screen region saved with SAVESCREEN(); the screen location may be the same or different.

Clipper syntax

RESTSCREEN(< expN1 > , < expN2 > , < expN3 > , < expN4 > ,
< expC >)

Result

RESTSCREEN() is converted to the dBASE IV command RESTORE SCREEN. However, unlike RESTSCREEN() which can restore a region of the screen, RESTORE SCREEN restores the entire screen.

SAVESCREEN()

Saves a screen region to a memory variable for later display.

Clipper syntax

SAVESCREEN(< expN1 > , < expN2 > , < expN3 > , < expN4 >)

Result

SAVESCREEN() is converted to the dBASE IV command SAVE SCREEN. However, unlike SAVECREEN() which can save a region of the screen, SAVE SCREEN saves the entire screen to a memory variable.

SCROLL()

Designates a screen region to scroll up, down, or blank out.

Clipper syntax

SCROLL(< expN1 > , < expN2 > , < expN3 > , < expN4 > , < expN5 >)

Result

SCROLL() is converted to a user-defined function that calls a binary routine.

SECONDS()

Determines the number of seconds elapsed since 12:00 A.M., midnight.

Clipper syntax

SECONDS()

Result

SECONDS() is converted to a user-defined function.

SELECT()

Returns the current work area number or the number of the work area for a specified alias.

Clipper syntax

SELECT()

Result

SELECT() is converted to a user-defined function.

SETCANCEL()

Toggles program termination with Alt-C *ON* or *OFF*.

Clipper syntax

SETCANCEL(< expL >)

Result

SETCANCEL() is converted to a user-defined function that toggles SET ESCAPE ON or OFF.

SETCOLOR()

Determines the current color setting and optionally changes it.

Clipper syntax

SETCOLOR(< expL >)

Result

SETCOLOR() is converted to the dBASE IV command SET COLOR TO. However, there is no function in dBASE IV that will return the current setting. When Step IVward encounters a SETCOLOR() function with no arguments, the command line commented out and a warning message is generated by Step IVward. The argument is converted to a null ("").

SETPRC()

Sets the internal PROW() and PCOL() values.

Clipper syntax

SETPRC(< expN1 > , < expN2 >)

Result

SETPRC() is replaced by a user-defined function which sets the dBASE IV system memory variables _plineno and _pcolno.

STRTRAN()

Searches and replaces within a character string.

Clipper syntax

STRTRAN(< expC1 > , < expC2 > [, < expC3 >]
[, < expN1 >][, < expN2 >])

Result

STRTRAN() is converted to a user-defined function.

SUBSTR()

SUBSTR() extracts a specified number of characters from a character expression or variable.

Clipper syntax

SUBSTR(< expC > , < string position > [, < number of characters >])

Result

Clipper supports negative values for the < string position > and < number of characters > arguments; dBASE IV does not. Step IVward comments out statements that include negative values.

TONE()

Sounds a speaker tone of a specified frequency and duration.

Clipper syntax

TONE(< expN1 > , < expN2 >)

Result

TONE() is converted to a user-defined function that uses the dBASE IV commands SET BELL TO and, ?? CHR(7).

USED()

Returns logical true if a file is in use in the current work area.

Result

USED() is converted to the dBASE IV expression LEN(DBF()) > 0.

WORD()

Converts numeric parameters of the CALL command from DOUBLE to INT.

Clipper syntax

WORD(< expN >)

Result

WORD() is converted to a user-defined function that returns the integer value of the passed argument.

Chapter 4: FoxBASE + to dBASE IV

Translation from FoxBASE + to dBASE IV is a very straight-forward process. These are the steps needed for a proper translation of a complete application.

1. Select **Input filename** on the **Setup** menu by pressing ↵.
2. Type the filename of your main application.
3. Select **Output filename** and press ↵.
4. Type the filename of your main application (the default is the same as your input filename).
5. Select **Process files tree** on the **Options** menu.
6. Select **Select language to convert from** on the **Translate** menu.
7. Select **FoxBASE +** as your input source language.
8. Now select **Begin translation.**

Your code is now translated directly to dBASE IV source code. There will be insertions of LOAD and CALL statements for the included binary files. Additional UDFs are included for the conversion of several of the FoxBASE + functions. Code lines that cannot be converted directly to either dBASE IV or a .bin file are inserted as comments with an appropriate warning or error message.

Essentials

This section covers some essential differences between FoxBASE + and dBASE IV. These are concepts and differences that Step IVward is unable to reconcile.

In order to process all data objects sufficiently, Step IVward makes two passes on your source code. In the first pass, all variable names, filenames, and other user-definable objects are parsed. The first pass is essential to differentiate between array names and user-defined functions. During the second pass, your code is translated.

In order to ensure that all objects in your application are correctly translated, you should set the **Process files tree** option of the **Options** menu ON when you translate an application. When this setting is ON, all files get the first pass before the translation begins. This will allow Step IVward to recognize the type of object used in a program other than the defining program.

Procedures and User-Defined Functions (UDFs)

FoxBASE + allows multiple PROCEDURE files to be open at the same time; dBASE IV allows only one PROCEDURE file to be open at a time. However, dBASE IV allows PROCEDURE and FUNCTIONS to be declared in program files. When declared in program files, a PROCEDURE's availability behaves similar to a memory variable. PROCEDUREs declared in a calling program are available to that program and all sub-programs called.

In FoxBASE +, there is not a seperate FUNCTION keyword. UDFs are declared with the PROCEDURE keyword. In dBASE IV, UDFs terminate with a RETURN < exp > command. If RETURN < exp > is present in a PROCEDURE, the dBASE IV compiler generates an error.

To translate FoxBASE + procedures and UDFs, Step IVward first determines if a routine is used as a procedure, a UFD, or both. The translation occurs as follows:

- **PROCEDURE only:** If the routine is used as a PROCEDURE only, it is left as a PROCEDURE. If there is an < exp > after the RETURN command, the < exp > is removed.
- **UDF only:** If the routine is used as a UDF only, Step IVward checks to see if any dBASE IV restricted commands are included. If no restricted commands exist, the PROCEDURE keyword is changed to FUNCTION. If restricted commands are present, the RETURN expression is removed and a previously declared variable is given the return value. Then in the offending command line, function is replaced with the memory variable and the procedure is called prior to executing the line.

- **Both PROCEDURE and UDF:** If the routine does double duty, Step IVward removes the RETURN expression. In the routine, a previously declared variable is assigned the return value. In command lines that contain function references, the reference is replaced with the memory variable and the procedure is called prior to executing the offending command.

Environmental Views

Though Step IVward is able to translate the CREATE VIEW command, dBASE IV does not support all of the FoxBASE + view functionality. dBASE IV does not maintain the current switch settings or path. If you need to save the switch settings, create a routine that saves each switch to a variable based on the SET() return value. Save the memory variables to a .mem file. The settings can then be restored by restoring the .mem file and resetting the switches. See SET() function the dBASE IV *Language Reference* manual for more details.

Relational and Logical Operators

Step IVward converts the FoxBASE + logical not (!) to dBASE IV .NOT. operator. The = = operator is translated to a = and a warning message is generated. To perform exact compares in dBASE IV, use the SET EXACT command.

Index Files

If INDEX = NDX is included in your FoxBASE + Config.db file, dBASE IV will flag this an an error. Additionally, dBASE IV will not be able to read FoxBASE + index files. Rebuild all index files with the dBASE IV INDEX ON command.

Program Flow Errors

If a function that cannot be translated is included in the < condition > of a program flow command, Step IVward generates an error message and either comments the line out or removes the condition forcing a compile-time error. The action depends on the following:

- **The condition is in a DO WHILE or IF statement:** the line is commented out.
- **The condition is in a CASE statement:** the < condition > is removed leaving CASE only.

FoxBASE + Commands

This section lists FoxBASE + commands that do not have analogs in dBASE IV and describes how Step IVward converts these commands.

@...BOX

Draws a box of specified characters to the output device.

Syntax

FoxBASE +

@ <row1>, <col1>, <row2>, <col2> BOX [<expC>]

dBASE IV syntax

@ <expN1>, <expN2> TO <expN3>, <expN4> <expC>

Result

In both languages, <expC> represents a border definition string, but the formats are different. In FoxBASE + the first eight characters define the box borders starting from the upper-left corner and then proceed clockwise (tl,t,tr,r,br,b,bl,l). The ninth character is a fill character. In dBASE IV, the order of the first eight is (t,b,l,r,tl,tr,bl,br). The values must be separated by commas. There is no fill character in dBASE IV. Step IVward drops the border definition string. This results in a single-line border.

If <expC> is a variable, Step IVward removes the variable name from the command line and generates a warning message. This is because the variable declaration cannot be translated automatically. To use a variable, change the order of the variable declaration and include the macro symbol (&).

Example

FoxBASE + code

```
frame = CHR(201)+CHR(205)+CHR(187)+CHR(186)+CHR(188)+ ; CHR(205)+CHR(200)+CHR(186)+CHR(176)
@ 1,0,24,79 BOX frame
```

dBASE IV code

```
frame = "CHR(205),CHR(205),CHR(186),CHR(186),CHR(201),CHR(187),CHR(200),CHR(188)"
@ 1,0 TO 24,79 &frame.
```


@...MENU

Creates a pop-up menu at a desired location on the screen.

Syntax

FoxBASE +

@ <row>, <col> MENU <array>, <expN1> [, <expN2>]
[TITLE <expC>]

dBASE IV

DEFINE POPUP <pop-up name> FROM <row1>, <col1>
[TO <row2>, <col2>]
DEFINE BAR <expN> OF <pop-up name> PROMPT <expC>
MESSAGE <expC>

Result

Step IVward converts @...MENU to a dBASE IV pop-up menu structure. The arguments are converted as follows:

<row>, <col>	The top left corner of the pop-up
<array>	<pop-up name>
<expN1>	Determines how many bars to create
<expN2>	The height of the pop-up (<row1> + <row2> + 2)
TITLE <expC>	Not used

@...PROMPT

Displays menu options at designated positions on the screen.

Syntax

FoxBASE +

@ <row>, <col> PROMPT <expC1> [MESSAGE <expC2>]

dBASE IV

DEFINE MENU <menu name>
DEFINE PAD <pad name> OF <menu name> PROMPT <expC>
[AT <row>, <col>] [MESSAGE <expC>]

or

DEFINE POPUP <pop-up name> FROM <row1>, <col1>
[TO <row2>, <col2>]
DEFINE BAR <expN> OF <pop-up name> PROMPT <expC>
MESSAGE <expC>

Result

Step IVward converts this command to either a pop-up or menu system, depending on the <col> values. If all of the <col> values are the same, Step IVward translates the menu to a pop-up menu system; otherwise, a menu bar system is created. The <col> values are evaluated starting with the first @...PROMPT command encountered until the first MENU TO command. The arguments are converted as follows:

Menu bar system

<row>, <col>	DEFINE PAD...AT coordinates
<expC1>	DEFINE PAD <pad name>
<expC2>	DEFINE PAD...MESSAGE <expC>

Pop-up menu system

<row>, <col>	The top left corner of the pop-up
<expC1>	DEFINE BAR...PROMPT <expC>
<expC2>	DEFINE BAR...MESSAGE <expC>

BROWSE

BROWSE is a full-screen menu-assisted command for editing and appending records in a database file.

Syntax

FoxBASE+

BROWSE [FIELDS <field list>][LOCK <expN>]
[FREEZE <expN>][NOFOLLOW] [NOMENU][NOAPPEND]
[WIDTH <expN>][NOMODIFY]

dBASE IV

BROWSE [FIELDS <field list>][LOCK <expN>]
[FREEZE <expN>][NOFOLLOW] [NOMENU][NOAPPEND]
[WIDTH <expN>][NOEDIT][NOAPPEND][NODELETE]

Result

dBASE IV recognizes all options of the FoxBASE+ BROWSE command except NOMODIFY. Step IVward replaces NOMODIFY with the dBASE IV options NOAPPEND, NOEDIT, and NODELETE.

CLEAR PROGRAM

Clears the compiled program buffer.

Result

dBASE IV does not support the CLEAR PROGRAM command. Step IVward comments the command line out and generates a warning message.

CLEAR PROMPT

CLEAR PROMPT releases all prompts displayed with @...PROMPT.

Result

dBASE IV does not support the CLEAR PROMPT command.

Step IVward comments this command line out and generates an error message.

CREATE VIEW

CREAT VIEW builds a view file of the current environment that is compatible with dBASE III PLUS.

Syntax

FoxBASE+

CREATE VIEW <.vue filename> FORM ENVIRONMENT [ALL]

dBASE IV

CREATE VIEW <.vue filename> FORM ENVIRONMENT

Result

If the ALL keyword is present, Step IVward removes it from the command line. See the "Environmental Views" in the "Essentials" section of this chapter for more information.

DIMENSION

Creates one- or two-dimensional arrays of memory variables.

Syntax

FoxBASE +

DIMENSION <array name1> (<rows1> {, <cols1>})
{, <array name2> (<rows1> {, <cols1>})...}

dBASE IV

DECLARE <array name1> [{<number of rows>,}
{<number of columns>}]{, <array name2> [{<number of rows>,}
{<number of columns>}]...)

In this paradigm, the curly braces indicate optional items. The square brackets are a required part of the DECLARE command sytnax.

Result

Step IVward changes DIMENSION to DECLARE, and changes parentheses around dimension arguments to square brackets ([]) both in the declare statement and in all references to the array(s) in other statements.

DIR

DIR displays directory information similar to that displayed by the DOS DIR command.

Syntax

FoxBASE +

DIR/DIRECTORY [<drive>] [<path>] [<skeleton>] [TO PRINT]

dBASE IV

DIRECTORY/DIR [[ON] <drive>:] [[LIKE] [<path>] <skeleton>]
LIST FILES [LIKE [<path>] <skeleton>]
[TO PRINTER/TO FILE <filename>]

Result

dBASE IV does not support the TO PRINT option in the DIR command. If TO PRINT is included in the command line, Step IVward changes the command to LIST FILES.

FLUSH

Flush all active buffers to disk.

Syntax

FoxBASE +

FLUSH

dBASE IV

dBASE IV has no corresponding command. The SET AUTOSAVE command in dBASE IV saves each record to disk after I/O has been performed on the record.

Result

Step IVward converts FLUSH to the following dBASE IV routine:

```
SET AUTOSAVE ON
SKIP 0
SET AUTOSAVE OFF
```

GATHER FROM

Replaces data from a memory variable array into the current database record.

Syntax

FoxBASE +

GATHER FROM <array> [FIELDS <field list>]

dBASE IV

dBASE IV does not support the GATHER FROM command.

Result

If there is not a <fieldlist> argument, Step IVward replaces GATHER FROM with the following code:

```
mindex = 1
DO WHILE mindex <= 255 .AND. TYPE('array[mindex']) <> 'U' ;
   .AND. LEN(FIELD(mindex)) ! 0
   mtemp = FIELD(mindex)
   REPLACE &mtemp WITH array[mindex]
   mindex = mindex + 1
ENDDO
```

If there is a <fieldlist>, Step IVward replaces GATHER with the following code:

```
REPLACE <field1> WITH array[1]
REPLACE <field2> WITH array[2]
REPLACE <field3> WITH array[3]
(and so on for each <field> in <fieldlist>)
```

INDEX

INDEX creates an index in which records from a database file are ordered alphabetically, chronologically, or numerically.

Syntax

FoxBASE +

INDEX ON <exp> TO <filename> [FOR <expL>]

dBASE IV

INDEX ON <exp> TO <filename>

Result

dBASE IV does not support the FOR argument. Step IVward removes the option from the command line and generates an error message.

KEYBOARD

KEYBOARD stuffs the keyboard buffer with a character string.

Syntax

FoxBASE +

KEYBOARD <expC>

dBASE IV

dBASE IV does not support the KEYBOARD command.

Result

Step IVward changes KEYBOARD to a call to a binary routine. In dBASE IV you can record a keystroke macro that, when replayed with the PLAY MACRO command, will stuff all recorded keys into the keyboard buffer.

MENU

MENU installs a menu into a menu bar.

Syntax

FoxBASE +

MENU <expN1>, <array>, <expN2> [,<expN3>]

dBASE IV

DEFINE BAR <expN> OF <pop-up name> PROMPT <expC>
MESSAGE <expC>
DEFINE POPUP <pop-up name> FROM <row1>, <col1>
[TO <row2>, <col2>]
ON PAD <pad name> OF <menu name> [ACTIVATE POPUP
<pop-up name>
ON SELECTION POPUP <command>

Result

MENU is translated into a dBASE IV pull-down menu. The arguments are converted as follows:

<expN1>	ON PAD <pad name>
<array>	<pop-up name>. The array elements become BAR PROMPT <expC>
<expN2>	The number of DEFINE BAR commands
<expN3>	DEFINE POPUP...TO <row2> where <row2> = (top left corner + <expN3> + 2)

MENU BAR

MENU BAR is used to install a menu bar <array> into the menu bar.

Syntax

FoxBASE+

MENU BAR <array>, <expN>

dBASE IV

DEFINE MENU <menu name>
DEFINE PAD <pad name> OF <menu name> PROMPT <expC1>
[AT <row>, <col>][MESSAGE <expC2>]

Result

Step IVward translates MENU BAR into a dBASE IV menu bar system. The arguments are translated as follows:

<array>	<menu name>
<array[<n>,1]>	PROMPT <expC1>
<array[<n>,2]>	MESSAGE <expC2>

MENU TO

MENU TO invokes a light-bar menu of currently defined prompts.

Syntax

FoxBASE+

MENU TO <memvar>

dBASE IV

ACTIVATE MENU <menu name> ACTIVATE POPUP <popup name>

Result

Step IVward converts MENU TO to the appropriate dBASE IV menu activation command. The new command is dependant on the type of menu system Step IVward built for the currently defined prompts.

ON KEY

ON KEY calls the execution of a program if a specified key is pressed.

Syntax

FoxBASE+

ON KEY = <expN> [<commands>]

dBASE IV

ON KEY [LABEL <key label name>] [<command>]

Result

Step IVward translates the FoxBASE+ <expN> to the corresponding dBASE IV key label name.

PUBLIC

Designates specified memory variables and arrays that you can use and alter in any dBASE program or subprogram.

Syntax

FoxBASE+

PUBLIC <memory variable list> [, fox]
PUBLIC <array>(<expN1>[,<expN2>])
[,<array>(<expN>[,<expN>])]...

dBASE IV

PUBLIC <memory variable list>/[ARRAY <array element list>]

Result

dBASE IV does not allow memory variables and arrays to be declared PUBLIC in the same command line. If arrays and variables are intermixed, Step IVward breaks the PUBLIC command into two separate command lines. If the fox option is present in the command line, Step IVward adds an additional command line to set fox to true.

READ MENU BAR TO

READ MENU BAR activates the menu structure defined by MENU BAR and MENU.

Syntax

FoxBASE +

READ MENU BAR TO <memvar1>, <memvar2> [SAVE]

dBASE IV

ACTIVATE MENU <menu name> [PAD <pad name>]

Result

Step IVward converts READ MENU BAR to an ACTIVATE MENU command. The value of <memvar1> is used to determine which pull-down menu to activate initially via the PAD <pad name> option. <memvar2> is not used during menu activation. After a selection has been made of the activated menu, a routine (defined in the translation of the MENU command) is executed. If SAVE is included, the routine saves the screen, deactivates the menu system, and restores the screen before continuing. The PAD() and BAR() values are used to define return values for <memvar1> and <memvar2>.

READ MENU TO

READ MENU TO activates the popup menu defined by the @...MENU command.

Syntax

FoxBASE +

READ MENU TO <memvar> [SAVE]

dBASE IV

ACTIVATE POPUP <popup name>

Result

Step IVward converts READ MENU TO to an ACTIVATE POPUP command. After a selection has been made of the activated menu, a routine (defined in the translation of the @...PROMPT command) is executed. If SAVE is included, the routine saves the screen, deactivates the menu, and restores the screen before continuing. The and BAR() value is used to define return values for <memvar>.

RESTORE SCREEN

Redisplays a screen saved with SAVE SCREEN.

Syntax

FoxBASE +

RESTORE SCREEN [FROM <memvar>]

dBASE IV

RESTORE SCREEN FROM <memvar>

In dBASE IV, the FROM option is required. If the option is not present, Step IVward saves the screen to a memory variable named s4-screen.

SAVE SCREEN

Saves a screen region for later display.

Syntax

FoxBASE +

SAVE SCREEN [TO <memvar>]

dBASE IV

SAVE SCREEN TO <memvar>

In dBASE IV, the TO option is required. If the option is not present, Step IVward saves the screen to a memory variable named s4-screen.

SCATTER

SCATTER stores data from the current database record into a memory variable array. If the array doesn't exist or is too small, it is created.

Syntax

FoxBASE +

SCATTER [FIELDS <fieldlist>] TO <array name>

dBASE IV

COPY TO ARRAY <array name> [FIELDS <field list>] [<scope>]

Result

The COPY TO ARRAY command copies the specified number of records to an already DECLAREd array. If the array does not exits when the COPY TO ARRAY command is issued, Step IVward includes code to define the array.

SCROLL

Scrolls a portion of the screen up or down.

Syntax

FoxBASE +

SCROLL <row1>, <col2>, <row2>, <col2>, <expN>

dBASE IV

dBASE IV does not support the SCROLL command.

Result

Step IVward converts SCROLL to a call to a binary file using the CALL() function.

SET CLEAR

Determines if the screen is cleared when SET FORMAT or QUIT is executed.

Result

dBASE IV does not support the SET CLEAR command. The screen is always cleared when SET FORMAT or QUIT is executed.

Step IVward comments this command line out and generates an error message.

SET RELATION

Syntax

FoxBASE +

SET RELATION TO [<expr> INTO <alias>][,[<expr> INTO <alias>]] [ADDITIVE]

dBASE IV

SET RELATION TO [<exp>] INTO <alias>[,<exp>] INTO <alias>]]...

Result

dBASE IV does not support the FoxBASE + ADDITIVE option. Step IVward removes ADDITIVE and generates a warning message.

FoxBASE + Functions

This section lists FoxBASE + functions that do not have analogs in dBASE IV and describes how Step IVward converts them.

ERROR()

Returns the number of the corresponding error message that caused an ON ERROR condition.

FoxBASE + syntax

ERROR()

Result

Step IVward issues a warning that the FoxBASE + error numbers are different from the dBASE IV error numbers.

FCOUNT()

Returns the number of fields in the current database file structure.

FoxBASE + syntax

FCOUNT()

Result

dBASE IV does not support the FCOUNT() function. Step IVward converts FCOUNT() to a user-defined function.

INKEY()

The INKEY() function returns an integer representing the most recent key pressed by the operator.

FoxBASE + syntax

INKEY([n])

Result

FoxBASE + supports fractional values for the optional numeric argument, dBASE IV does not. If a fraction or a memory variable is detected, Step IVward changes the statement to INKEY(ROUND(n)).

RECNO()

RECNO() returns the current record number from the database file.

FoxBASE + syntax

RECNO(< expN >)

Result

The dBASE IV RECNO() function is the same as the FoxBASE + function except when an argument is present. An argument in the dBASE IV RECNO() function specifies the alias of the database file being queried.

When the FoxBASE + RECNO() function argument is 0, FoxBASE + uses soft seek logic after an unsuccessful SEEK to determine the record number to return. When the argument is 0, Step IVward generates a warning message indicating that the program logic should be reworked to use the dBASE IV SET NEAR command.

MESSAGE()

Returns the message associated with the last ERROR() generated.

FoxBASE + syntax

MESSAGE()

Result

No conversion is performed by Step IVward. However, the messages returned by dBASE IV are different than the messages returned by FoxBASE + .

READKEY()

The READKEY() function returns an integer for the key pressed to exit a full-screen command or light-bar menu.

FoxBASE + syntax

READKEY()

Result

dBASE IV READKEY() does not support exit keys from bar menus. In dBASE IV, use the LASTKEY() fucntion can be used to determine the key pressed to exit a menu. Step IVward makes no conversion.

SELECT()

SELECT() returns the current work area number.

FoxBASE + syntax

SELECT()

The dBASE IV SELECT() function returns the last available work area. Step IVward converts SELECT() to a user-defined function.

SYS()

Returns various system values as character strings depending on the value of the first argument.

FoxBASE + syntax

SYS(< expN1 > [, < expN2 > [, < expN3 >]])

Result

dBASE IV does not support the SYS() function. Instead, Step IVward provides a program file that includes the functionality of SYS().

Command lines that include the SYS() function are prefaced with the command DO Sys WITH < expN1 > , < exp1 > , < exp2 > and SYS() is replaced with the memory variable m_retval. If the second and third arguments are not present in the original function, 0 is supplied for < expN2 > and < expN3 > . This is because dBASE IV requires a fixed number of parameters to be passed to a PARAMETERS statement.

The output provided by Sys.prg for each argument of SYS() is described in table 4-1.

Table 4-1 Sys.prg return values

Argument	Description	Sys.prg returns
0	Returns the machine name and network machine number as defined on your network	A character string containing the name of the network. The machine number is not provided.
1	Returns current system date as a Julian day number	A seven-character long string

(continued)

Table 4-1 Sys.prg return values (*continued*)

Argument	Description	Sys.prg returns
2	Returns the number of seconds elapsed since midnight	A five-character long string
3	Returns a unique temporary filename	A unique 8-character long string
5	Returns default disk drive	The drive letter as a character string. The colon is included.
6	Returns current PRINT device as set by SET PRINTER TO	A null string
7 [,w]	Returns the name of the current FORMAT file	A null string
9	Returns the FoxBASE + serial number	A null string
10, d	Converts the numeric day number to a character string	An eight-character string
11, s	Converts a date or character string in date format to a Julian day number	A seven-character long string
12	Returns the amount of available memory in bytes	A character string based on the MEMORY() function
13	Returns a character string indicating the status of the printer	A character string based on the PRINTSTATUS() function
14, n [,w]	Returns the index expression for index file number *n* in the current select area or in select area number w if *w* is specified.	A character string based on the KEY() function
15	Used for indexing on fields containing diacritical marks (for European users)	Returns the input string

(*continued*)

Table 4-1 Sys.prg return values (*continued*)

Argument	Description	Sys.prg returns
16 [,expN]	Returns the name of the program or procedure currently being executed or < expN > levels back	A character string based on the PROGRAM() function if the argument is omitted; otherwise, a null string is returned.
17	Returns the type of processor in use (8086/88, 80286 or 80386)	A null string
18	Returns the name of the current GET memvar or field when an ON KEY routine was triggered while a READ was active	Returns a character string based on the VARREAD() function
100	Returns the current SET CONSOLE status	A character string based on the expression SET("CONSOLE")
101	Returns the current SET DEVICE TO status	A null string
102	Returns the current SET PRINTER TO status	A character string based on the expression SET("PRINTER")
103	Returns the current SET TALK status	A character string based on the expression SET("TALK")
2000, c [,1]	Returns a filename matching the wildcard *c*	A character string containing the filename
2001, c [,1]	Returns SET command c's setting	A valid dBASE IV switch setting in a character string; otherwise, a null string
2002, [,1]	Turns the cursor off if 1 is passed, on if 0 is passed	Sets the cursor status and returns a null string
2003	Returns the current directory on the default drive	A character string
2004	Returns the home directory where FoxBASE + is executing	A null string

UPDATED()

UPDATED() determines if data were changed in last full screen command.

FoxBASE + syntax

UPDATED()

Result

Step IVward changes UPDATED() to the dBASE IV expression READKEY() > = 256.

VERSION()

Returns a character string representing the version of FoxBASE + in use.

FoxBASE + syntax

VERSION()

Result

Although dBASE IV also has the VERSION() function, the values returned are obviously different. Step IVward generates a warning message when VERSION() is encountered.

Chapter 5: Quicksilver to dBASE IV

Translation from Quicksilver to dBASE IV is a very straight-forward process. These are the steps needed for a proper translation of a complete application.

1. Select **Input filename** on the **Setup** menu by pressing ↵.
2. Type the filename of your main application.
3. Select **Output filename** and press ↵.
4. Type the filename of your main application (the default is the same as your input filename).
5. Select **Process files tree** on the **Options** menu.
6. Select **Select language to convert from** on the **Translate** menu.
7. Select **Quicksilver** as your input source language.
8. Now select **Begin translation.**

Your code is now translated directly to dBASE IV source code. There will be insertions of LOAD and CALL statements for the included .bin files. Additional UDFs are included for the conversion of several of the Quicksilver functions. Code lines that cannot be converted directly to either dBASE IV or a .bin file are inserted as comments with an appropriate warning or error message.

Essentials

This section covers some essential differences between Quicksilver and dBASE IV. These are concepts and differences that Step IVward is unable to reconcile.

Commented Lines

Quicksilver ignores commented lines when the command lines begin with &&\ or *\. Step IVward removes these comments and translates the command.

The special Quicksilver switches *QSON and *QSOFF are ignored and left as comments. However, commands line between these commands are commented out.

Program Flow Errors

If a function that cannot be translated is included in the < condition > of a program flow command, Step IVward generates an error message and either comments the line out, or removes the < condition > causing a compile time error. The action is as follows:

- **The condition is in a DO WHILE or IF statement:** the command line is commented out.
- **The condition is in a CASE statement:** the < condition > is removed.

Quicksilver Diamond Release

Step IVward translates the Quicksilver Diamond Release. This means that commands and functions dependent on the supplemental Networker Plus™ software are not converted to dBASE IV code. When Step IVward encounters one of these keywords, the command line is commented out and an error message is generated. The Networker Plus dependent commands and functions that are not translated are as follows:

ACTIVE()
DOWNSCROLL
EXECUTE()
FLAG
MSGQUEUE()
ON NETERROR
RECEIVE()
RECEIVE SCREEN
SEND MESSAGE
SEND SCREEN
SEND TASK
SET MSGBELL
SET MEGQUEUE
SET MSGWIN
SET SNOOP
SET TASKQUEUE
TASKQUEUE()
USERCOUNT()
USERNAME()
USERNO()
WHOAMI()
WHOHASIT()
WHOSENTIT()

Windows

Quicksilver and dBASE IV both support windows; however, the manner of support is very different. In order to provide Quicksilver like support, Step IVward adds an array to the root module of your routine. This array is used to store the window names. All translated windowing commands then access this array for the necessary information.

dBASE IV has a maximum limit of 20 defined at one time. You may have to modify your program to release windows that are not being used.

AUTOMEM Variables

Quicksilver allows the addition of the keyword AUTOMEM to each of the following standard dBASE commands:

USE...AUTOMEM
CLEAR AUTOMEM
STORE AUTOMEM
APPEND AUTOMEM
INSERT AUTOMEM
REPLACE AUTOMEM
RELEASE AUTOMEM

Conversion of these will be as follows:

USE ... AUTOMEM

Step IVward removes the statement and inserts code to initialize memory variables with the same names as the current database file fields. The values of the current record's fields are stored into the matching variables.

CLEAR AUTOMEM

Step IVward inserts code to initialize memory variables based on the database file fields. The length and data type match the corresponding fields.

STORE AUTOMEM

Step IVward inserts code to store the contents all of the fields in the current record to memory variables. The memory variable names are the same as the field names.

APPEND AUTOMEM

Step IVward inserts code to add a new record the database file. The contents of the new record are then replaced with values from memory variables. The memory variable names are the same as the field names.

INSERT AUTOMEM

Step IVward inserts code to insert a blank record. The contents of the record are then replaced with values from memory variables. The memory variable names are the same as the field names.

REPLACE AUTOMEM

Step IVward inserts code to replace the field contents in the current record with the contents of memory variables. The memory variable names are the same as the field names.

RELEASE AUTOMEM

Step IVward inserts code to release from memory, all memory variables whose names match the field names in the current database file.

Commands

This section lists Quicksilver commands that do not have analogs in dBASE IV and describes how Step IVward converts these commands.

@

@ displays or accepts information in a specified format at a given set of screen coordinates.

Syntax

Quicksilver

@...GET...[HELP <expC>]

dBASE IV

@...GET...[MESSAGE <expC>]

Result

The Quicksilver HELP option displays a help message given by <expC> (up to 254 characters long) if the user presses **F1 Help** while in the corresponding GET field. Step IVward converts HELP to MESSAGE. In dBASE IV, the MESSAGE expression is displayed whenever the cursor moves to the GET field.

Context sensitive help systems can be written in dBASE IV using the ON KEY command in conjunction with the VARREAD() function.

CCALL

Allows calls to compiled C language routines.

Result

dBASE IV does not support the CCALL command. Command lines that include CCALL are commented out and an error messages is generated.

CLEAR

Clears the screen or window and, optionally fills the cleared area with a specified character.

Syntax

Quicksilver

CLEAR [CHARACTER < expC >]

dBASE IV

CLEAR

Result

dBASE IV does not support the CHARACTER argument. Step IVward removes the CHARACTER argument and generates an error message.

CLEAR SENSERANGE

Restarts edit sensing in the current range of records in the active database file marked for edit sensing with SET SENSERANGE.

Result

dBASE IV does not support the CLEAR SENSERANGE command. Step IVward comments the command line out and generates an error message.

CREATE

Creates an empty structure-extended database file.

Syntax

Quicksilver

CREATE < filename >

dBASE IV

dBASE IV does not create an empty structure extended database file. Instead, the dBASE IV CREATE command gives you access to the database file design.

Result

Step IVward converts CREATE to a call to a procedure that generates an empty structure extended file.

DIMENSION

DIMENSION declares an array of memory variables.

Syntax

Quicksilver

DIMENSION < array name > [< expN list >], < array name >
[< expN list >]...

dBASE IV

DECLARE < array name1 > [{ < number of rows > ,}
{ < number of columns > }][, < array name2 > [{ < number of rows > ,}
< number of columns > }]...)

Result

Step IVward converts DIMENSION to the dBASE IV command DECLARE.

DOSINT

DOSINT allows access to DOS interrupts.

Result

dBASE IV does not support the DOSINT command. Step IVward comments this commend out and generates an error message.

DOWNSCROLL

Scrolls the current screen text (or current window area) down a specified number of lines.

Result

dBASE IV does not support the DOWNSCROLL command. Step IVward comments this command line out and generates an error message.

EXPORT

Copies the open database file to a Ventura™ file format.

Result

dBASE IV does not support the Ventura file type. Step IVward comments this command line out and generates an error message.

FOR...NEXT

A structured programming command that allows statements to be repeated a fixed number of times.

Syntax

Quicksilver

```
FOR <memvar> = <expN1> TO <expN2> [STEP <expN3>]
   <Commands>
  [BREAK]
   <Commands>
NEXT
```

dBASE IV

```
<memvar> = <expN1>
DO WHILE <memvar> <= <expN2>
   <commands>
  [EXIT]
   <commands>
   <memvar> = <memvar> + <expN3>
ENDDO
```

Result

Step IVward converts FOR...NEXT structures to DO WHILE...ENDDO constructs. Any BREAK commands are converted to EXIT. If the STEP argument is negative, the DO WHILE test is changed to >=. If the argument is an expression, Step IVward cannot determine the sign of the result; you will have to change the operators manually if the result is to be negative.

GRAPH FORM

Displays a graph using data from up to 20 records in the current database file.

Result

dBASE IV does not support the GRAPH FROM command. Step IVward comments this command out and generates an error message.

ON EVENT

Specifies a command to be executed when an event is detected by SET EVENT.

Result

dBASE IV does not support the ON EVENT command. Step IVward comments this command out and generates an error message.

ON KEY

Specifies a command to be executed when a specified key or any key is pressed.

Syntax

Quicksilver

```
ON KEY [<key>] [<command>]
```

dBASE IV

```
ON KEY [LABEL <key label name>][<command>]
```

Result

The Quicksilver ON KEY command is identical to the dBASE IV command. If the <key> option is included, Step IVward converts the command to the dBASE IV ON KEY LABEL command. Quicksilver key codes are converted to the matching dBASE IV key label names.

OUT

Sends the result of a numeric expression to a specified port.

Result

dBASE IV does not support the OUT command. Step IVward comments this command out and generates an error message.

RESTORE GRAPH

Displays a graph created by the GRAPH FORM command.

Result

dBASE IV does not support the RESTORE GRAPH command. Step IVward comments this command out and generates an error message.

SET AUTOLOCK

Controls whether a program must obtain a record lock prior to changing the record.

Syntax

Quicksilver

SET AUTOLOCK on/OFF

dBASE IV

SET LOCK ON/off

Result

Step IVward converts SET AUTOLOCK to SET LOCK.

SET COLOR TO

Sets the color selection and specifies a fill character for the background.

Syntax

Quicksilver

SET COLOR TO [[< standard >] [, < enhanced >] [, < border >]]
[CHARACTER < expC >]

dBASE IV

SET COLOR TO [[< standard >] [, [< enhanced >] [, [< border >]]]]

Result

dBASE IV does not support the CHARACTER argument. Step IVward removes the CHARACTER argument and generates a warning message.

SET CURSORMOVE

Determines whether the cursor is moved while the screen is updated.

Result

dBASE IV does not support the SET CURSOR MOVE command. Step IVward comments this command out and generates an error message.

SET DBF

Specifies a secondary directory search path for database files.

Result

dBASE IV does not support the SET DBF command. Also, Step IVward does not detect if DBF is included in the DOS SET variable list. Step IVward comments this command out and generates an error message.

SET DEVICE

Specifies whether @...SAY commands are routed to the screen, the printer, or a file.

Syntax

Quicksilver

SET DEVICE TO [SCREEN/PRINT/ALTERNATE]

dBASE IV

SET DEVICE TO [SCREEN/PRINTER/FILE < filename >]

Result

Step IVward converts the ALTERNATE keyword to FILE.

SET EDITOR

Specifies the editor for editing memo fields.

Result

In dBASE IV, the editor can be defined with the WP = option of Config.db. Step IVward comments this command out and generates an error message.

SET EVENT

Specifies a routine that is continuously invoked by Quicksilver to check for an event occurrence.

Result

dBASE IV does not support SET EVENT. Step IVward comments this command out and generates an error message.

SET FEED

Specifies whether a form feed is sent to the printer when the row of an @...SAY is less than the row of a previous @...SAY.

Result

dBASE IV does not support SET FEED. dBASE IV always sends a form feed to the printer when the current row value is less than the previous row value. Step IVward comments this command out and generates an error message.

SET FLAG

Set a specified flag on or off.

Result

dBASE IV does not support SET FLAG. Step IVward comments this command out and generates an error message.

SET FUNCTION

SET FUNCTION programs a function key.

Syntax

Quicksilver

SET FUNCTION <exp> / <expN> TO <expC>

dBASE IV

SET FUNCTION <expN> / <expC> / <key label> TO <expC>

Result

Unlike dBASE IV, Quicksilver allows the **F1** key to be reprogrammed. If **F1** is specified, the command line is commented out and an error message is generated.

SET MULTIUSER

Controls whether a workstation accesses files in single-user or multi-user mode.

Syntax

Quicksilver

SET MULTIUSER ON / off

dBASE IV

SET EXCLUSIVE on/OFF SET LOCK ON/off

Result

SET MULTIUSER is converted to SET EXCLUSIVE and SET LOCK.

SET NDX

Specifies a secondary directory search path for index files.

Result

dBASE IV does not support the SET NDX command. Step IVward does not detect if DBF is included in the DOS SET list. Step IVward comments this command out and generates an error message.

SET PRINTER

SET PRINTER TO redirects output to a local printing device, or to a file.

Syntax

Quicksilver

SET PRINTER TO [<device> / <filename>]

dBASE IV

SET PRINTER TO <DOS device>
SET PRINTER TO FILE <filename>

Result

If a filename is used instead of a DOS device, Step IVward inserts the keyword "FILE".

SET RETRACE

When SET RETRACE in ON, Quicksilver waits for a vertical retrace before writing to the screen.

Result

dBASE IV does not support the SET RETRACE command. Step IVward comments this command out and generates an error message.

SET SENSERANGE

Specifies the range of records in the active database files for edit sensing.

Result

dBASE IV does not support the SET SENSERANGE command. Step IVward comments this command out and generates an error message.

SET TIME

SET TIME TO sets the system time.

Syntax

Quicksilver

SET TIME TO <hh:mm:ss>

dBASE IV

dBASE IV does not support SET TIME.

Result

Step IVward converts SET TIME to RUN TIME hh:mm:ss!.

SET USERHELP

SET USERHELP ON/off determines whether user help messages are displayed when a user presses **F1 Help** during a READ.

SET USERHELP TO defines the window used to display user help messages during a READ.

Result

dBASE IV does not support SET USERHELP. Step IVward comments these commands out and generates an error message.

In dBASE IV program files, the **F1** key can be disabled from accessing the system help files with the command ON KEY F1 ??. Also, context sensitive help systems can be written in dBASE IV using the ON KEY command in conjunction with the VARREAD() function.

SKIP

SKIP moves the record pointer forward or backward in a database file.

Syntax

Quicksilver

SKIP [<expN>] [ALIAS <alias>]

dBASE IV

SKIP [<expN> [IN <alias>]

Result

Step IVward converts the keyword ALIAS to IN if it is present in the command line.

SLEEP

Pauses program execution for, or until, a specified time and date.

Syntax

Quicksilver

SLEEP < expN > / UNTIL < expC1 > [, < expC2 >]

dBASE IV

INKEY(< expN >)

Result

If the UNTIL option is not present, Step IVward forward changes the command line to m_temp = INKEY(< expN >). Otherwise, SLEEP is converted to a DO WHILE loop whose condition is TIME() < > < expC1 > .AND. DATE() < > < expC2 > .

UPSCROLL

Scrolls the current screen text (or current window area) up a specified number of lines.

Syntax

Quicksilver

UPSCROLL < expN >

dBASE IV

dBASE IV does not support the UPSCROLL command.

Result

Step IVward comments this command out and generates an error message. However, specific windows can be moved in dBASE IV with the MOVE WINDOW command.

WABANDON

Deselects a window or group of windows that were previously selected using WSELECT; the window(s) remain on the screen.

Syntax

Quicksilver

WABANDON [< area number > [TO < area number >]]/[ALL]

dBASE IV

dBASE IV does not support WABANDON.

Result

Step IVward translates WABANDON to various dBASE IV commands. The commands may included a combination of ACTIVATE/RESTORE/SAVE SCREEN and ACTIVATE/DEACTIVATE WINDOW.

WCLOSE

Deselects a window or group of windows that were previously selected using WSELECT; the previous contents of the area(s) covered by the window(s) are redisplayed.

Syntax

Quicksilver

WCLOSE [[< area number > [TO < area number >]] / [ALL]]

dBASE IV

DEACTIVATE WINDOW < window name list > /ALL

Result

Step IVward translates WCLOSE to a procedure based on the DEACTIVATE WINDOW command.

WCOPY

Copies the currently-selected frame and window to another window area.

Result

dBASE IV does not support WCOPY TO. Step IVward comments this command out and generates an error message.

WDISPLAY

Clears the contents of the specified window(s) and redisplays the frame(s).

Syntax

Quicksilver

WDISPLAY [[< area number > [TO < area number >]] / [ALL]]

dBASE IV

dBASE IV does not support WDISPLAY.

Result

Step IVward translates WDISPLAY to a procedure based on the ACTIVATE WINDOW and DEACTIVATE WINDOW commands.

WMOVE

Moves the currently selected window to another screen location.

Syntax

Quicksilver

WMOVE TO [< new top row >][, < new left col >]

dBASE IV

MOVE WINDOW < window name > TO < row > , < col >

Result

If both coordinates are provided, Step IVward translates WMOVE to the MOVE WINDOW...TO command. If either or both coordinates are missing, Step IVward comments the line out and generates an error message.

WRELEASE

Releases windows from memory.

Syntax

Quicksilver

WRELEASE < window name > /ALL

dBASE IV

RELEASE WINDOWS < window name list > CLEAR WINDOWS

Result

WRELEASE is converted to RELEASE WINDOWS unless the ALL option is included. IF the ALL option is included, WRELEASE in converted to CLEAR WINDOWS.

WRESTORE

Restores a window image from a memory buffer or a screen file.

Syntax

Quicksilver

WRESTORE [FROM < screen file >]

dBASE IV

RESTORE WINDOW ALL FROM < filename >

Result

Step IVward translates WRESTORE FROM to RESTORE WINDOW ALL FROM. If the FROM option is not included, Step IVward comments the line out and generates an error message.

WSAVE

Saves a window image to a memory buffer or a screen file.

Syntax

Quicksilver

WSAVE [TO <screen file>]

dBASE IV

SAVE WINDOW <window name> TO <filename>

Result

Step IVward translates WSAVE to the SAVE WINDOW TO command, regardless of the presence of the WSAVE TO option. If the TO option is not present, Step IVward uses the window name as the filename.

WSELECT

Selects the window area to be active.

Syntax

Quicksilver

WSELECT <area number>

dBASE IV

dBASE IV does not support the WSELECT command.

Result

Step IVward translates Quicksilver windowing commands by building an array to maintain the window names and select areas. WSELECT is converted to code that determines if there is a window defined in the selected area. If a window exists, an ACTIVATE WINDOW command is issued. If no window is defined for the selected area, ACTIVATE SCREEN is issued.

WSET FRAME

Determines whether a border is displayed around windows.

Syntax

Quicksilver

WSET FRAME ON/off

dBASE IV

SET BORDER TO [NONE]

Result

If the WSET FRAME argument is ON, the command line is converted to SET BORDER TO. SET BORDER TO with no arguments generates a single line box. When the WSET FRAME argument is OFF, the command line is translated to SET BORDER TO NONE. A frames appearance with a window is determined by the border definition at the time the window was created.

WSET SIZE

Erases and sets the size of the current window.

Syntax

Quicksilver

WSET SIZE TO <top row>, <left col>, <bottom row>, <right col>

dBASE IV

DEFINE WINDOW <window name> FROM <row1>, <col1> TO <row2>, <col2>
ACTIVATE WINDOW <window name>

Result

Step IVward translates WSET SIZE to a routine based on the DEFINE WINDOW and ACTIVATE WINDOW commands. If the specified window is active at the time, a DEACTIVATE WINDOW command is also issued.

WSET TITLE

Sets a title to appear in the top border of the currently selected window.

Result

dBASE IV does not support the WSET TITLE command. Step IVward comments this command out and generates an error message.

WSET WINDOW

Defines a window for display.

Syntax

Quicksilver

WSET WINDOW <window name> TO <tr,lc,br,rc>
[CHARACTER <background char>]
[COLOR [<standard>] [[,<enhanced>] [,<border>]]]
[FRAME <tl,t,tr,ls,rs,bl,b,br>/DOUBLE]

dBASE IV

DEFINE WINDOW <window name> FROM <tr,lc> TO <br,rc>
[COLOR [<standard>] [[,<enhanced>] [,<border>]]]
[<border definition string>/DOUBLE]

Result

dBASE IV does not support the CHARACTER option. Step IVward removes the CHARACTER argument form the command line and generates an error message.

WSET WINFILE

Opens a master windows file from which saved windows may be restored.

Result

dBASE IV does not support the WSET WINFILE command. Step IVward comments this command out and generates an error message.

WUSE

Opens a window (previously defined using WSET WINDOW) in the current window select area, closing any window.

Syntax

Quicksilver

WUSE [<window name> [TITLE <string> [LEFT/CENTER/RIGHT]]]

dBASE IV

ACTIVATE WINDOW <window name>
DEACTIVATE WINDOW <window name>

Result

WUSE is translated to a routine utilizing the ACTIVATE WINDOW and DEACTIVATE WINDOW commands. The TITLE option is not translated by Step IVward.

Quicksilver Functions

This section lists Quicksilver functions that do not have analogs in dBASE IV and describes how Step IVward converts them.

BITSET()

Indicates if a specified bit of a number's binary representation is on or off.

Quicksilver syntax

BITSET(<memvar>, <position>)

Result

dBASE IV does not support BITSET(). Step IVward converts BITSET() to a user-defined function.

CEIL()

The CEILING() function calculates and returns the smallest integer that is greater than or equal to the value specified in the numeric expression.

Quicksilver syntax

CEIL(<expN>)

Result

CEIL() does not need to be translated because it is considered the short form of the dBASE IV CEILING() function.

CENTER()

Returns the specified character string centered in a line of the specified length.

Quicksilver syntax

CENTER(<expC>, [,<expN>]

Result

dBASE IV does not support CENTER(). Step IVward converts CENTER() to a user-defined function.

DESCEND()

DESCEND returns a character string whose character's ASCII values are 255 less than the value passed.

Quicksilver syntax

DESCEND(< expC >)

Result

dBASE IV does not support DESCEND(). Step IVward converts DESCEND() to a user-defined function.

EMPTY()

Determines if an expression evaluates to an empty value.

Quicksilver syntax

EMPTY(< exp >)

Result

dBASE IV does not support EMPTY(). Step IVward converts EMPTY() to a user-defined function. The user-defined function includes a CASE statement that compares the TYPE() of the input. EMPTY() returns true for the following conditions:

Data type	Test
Character	All spaces
Numeric	Value of zero
Logical	Value of false
Date	The string conversion is all spaces
Memo	MLINES() = 0

ERROR()

Returns the number of the corresponding error message that caused an ON ERROR or ON NETERROR condition.

Quicksilver syntax

ERROR()

Result

Step IVward issues a warning that the Quicksilver error numbers are different than the dBASE IV error numbers.

FCOL()

Returns the current column position in a file.

Result

Step IVward comments out command lines that contain this function and generates an error message.

FROW()

Returns the current row position in a file.

Result

Step IVward comments out command lines that contain this function and generates an error message.

HTOI()

Returns the decimal value of the specified hex number.

Quicksilver syntax

HTOI(< expC >)

Result

dBASE IV does not support HTOI(). Step IVward converts HTOI() to a user-defined function.

IN()

Returns a numeric value from the specified input port.

Result

Step IVward comments out command lines that contain this function and generates an error message.

ITOH()

Returns a character string containing the hexadecimal representation of the value specified.

Quicksilver syntax

ITOH(< expN1 > [, < expN2 >]

Result

dBASE IV does not support ITOH(). Step IVward converts ITOH() to a user-defined function.

MEMORY()

Returns the amount of memory available.

Quicksilver syntax

MEMORY()

Result

In dBASE IV, MEMORY() returns a value in Ks. Step IVward converts this function to MEMORY() / 1024.

MESSAGE()

Returns the message of the corresponding error that caused an ON ERROR or ON NETERROR condition.

Quicksilver syntax

MESSAGE()

Result

Step IVward issues a warning that the Quicksilver error messages are different than the dBASE IV error messages.

PRINTER()

Determines if the specified printer is online.

Quicksilver syntax

PRINTER(< port name >)

Result

Unlike the Quicksilver PRINTER() function, the dBASE IV PRINTSTATUS() function returns the status of the currently selected print device. Step IVward comments this command line out and generates an error message.

PROPER()

Returns a character string where each word of the input has its first letter set to upper case and the remaining letters set to lower case.

Quicksilver syntax

PROPER(< expC >)

Result

dBASE IV does not support PROPER(). Step IVward converts PROPER() to a user-defined function.

SELECT()

Returns the current work area number.

Quicksilver syntax

SELECT()

Result

The dBASE IV SELECT() is not the same as the Quicksilver SELECT(). Step IVward converts SELECT() to a user-defined function.

SENSERANGE()

Returns the last user name that altered data in the range of records specified in the last SET SENSERANGE.

Result

Step IVward comments out commands that contain this function and generates an error message.

SINKEY()

SINKEY() returns a character string representing the most recent key pressed by the operator.

Quicksilver syntax

SINKEY()

Result

dBASE IV does not support SINKEY(). Step IVward converts SINKEY() to a user-defined function.

VERSION()

Returns a character string representing the name and the version number of the Quicksilver compiler.

Quicksilver syntax

VERSION()

Result

Although dBASE IV also has the VERSION() function, the values returned are obviously different. Step IVward generates a warning message when VERSION() is encountered.

WACTIVE()

Indicates if a window is in use in the current window select area.

Quicksilver syntax

WACTIVE()

Result

Step IVward converts WACTIVE() to a UDF that returns the select area status.

WSELECT()

Returns the number of the current window select area.

Quicksilver syntax

WSELECT().

Result

Step IVward converts WSELECT() to a UDF that returns the select area status.

Chapter 6: Binary Files

The binary files in Step IVward are assembly-language files compiled under the Microsoft® Macro Assembler. These modules are written in 8086/8088 assembler code that can be loaded by dBASE IV. You may also create your own binary files using the DOS Macro Assembler.

To create a binary file:

1. Assemble your source code:

   masm < source > < target > null null

2. Link object files and create an executable .exe file:

   link < target > null

3. Create a binary (.bin) file from the executable file:

   exe2bin < target >

Here is an example of using a binary file that displays the current drive:

```
LOAD Getdrive
m_drive = " "
CALL Getdrive WITH mdrive
? "The current drive is ", m_drive
```

or

```
LOAD Getdrive
m_drive = " "
? "The current drive is ",CALL("Getdrive",m_drive).
```

You may find that there may be binary files that you want to group together at LOAD time, because of the translation. Keep in mind that dBASE IV can load only 16 .bin files at any one time.

For a more detailed discussion of using binary files with dBASE, see *dBASE POWER: Building, Creating & Using Programming Tools* by Olympia, Freeland and Wallin, published by Ashton-Tate, 1988.

Description of Binary Files

Following is a list of binary files included with Step IVward. These are needed for conversion of several of the functions and commands that dBASE IV does not support. Some of the binary files are called from procedures and UDFs of the same name. See Chapter 7 for a description of the supplemental program files.

Adir.bin

Fills a series array. This is information from a DIR.

Parameter

A character string whose first characters are a number denoting the type of call. The string must be padded with spaces, see the samples in the table for minimum padding lengths.

Type	Description	Sample
11	Find first file matching skeleton	"11" + mskel + SPACE(12)
2	Find next file matching skeleton	"2" + SPACE(12)
3	Return size	"3" + SPACE(7)
4	Return date	"4" + SPACE(7)
5	Return time	"5" + SPACE(7)
6	Return attributes	"6" + SPACE(3)

Returns

A character string containing the requested data.

Curdir.bin

Displays the current directory.

Parameter

A character buffer whose length is 65. For example, SPACE(65).

Returns

The current directory.

Example

```
? TRIM(CALL("Curdir",SPACE(65))
```

Curson.bin

Turns the cursor on. There are no parameters or return values.

Cursoff.bin

Turns the cursor off. There are no parameters or return values.

Fclose.bin

Closes the corresponding DOS file to the handle.

Parameter

A five character string containing the file handle.

Returns

True (.T.) if successful, false (.F.) if an error occurred.

Fcreate.bin

Creates a new DOS file. It can also truncate an existing file to a length of zero.

Parameter

A character string whose first character is a number denoting the file open mode. The string length must be at least 4. The access modes are as follows:

0	Normal
1	Read only
2	Hidden
3	System

Returns

The file handle as a numeric value. A −1 is returned if an error occurred.

Fread.bin

Fills character memory variables with characters from a DOS file.

Parameter

A character string whose first five characters are the file handle. The next character is the number of characters to read. The rest of the string is space padded with the number of characters to read minus 5; unless the number of characters to read is less than 5, then no padding is needed.

Returns

A character string containing the text.

Example

```
fhandle = FOPEN("Temp.txt")
fcount = 128
m_buffer = STR(fhandle) + CHR(fcount)
IF fcount < 5
   m_buffer = m_buffer + SPACE(fcount - 5)
ENDIF
newtext = CALL('FREAD', m_buffer)
```

Fopen.bin

Opens a DOS file directly.

Parameter

A character string whose first character is a number denoting the file open mode. The string length must be at least 4. The open modes are as follows:

1	Read only
2	Write only
3	Read/Write

Returns

The file handle as a numeric value. A −1 is returned if an error occurred.

Fseek.bin

Sets the DOS file pointer to a new position.

Parameters

A character string whose first five characters are the file handle. The next character five characters are the number of bytes to move the pointer. The last character denotes the method of moving the pointer. The valid choices for the method are as follows:

1	From the beginning of the file
2	From the current pointer position
3	From the end of the file

Returns

A numeric value representing the new position from the beginning of the file. If an error occurred, −1 is returned.

Fwrite.bin

Fills a DOS-specified file with a buffer variable.

Parameters

A character string whose first five characters are the file handle. The rest of the string contains the text to write to the file.

Returns

The number of bytes successfully read. If an error occurred, −1 is returned.

Keyboard.bin

Stuffs characters into the keyboard buffer.

Parameter

A character string containing the characters to stuff into the keyboard buffer.

Returns

Nothing.

Netname.bin

Returns the network name.

Parameter

A character string whose length is 18. The last 3 characters must be "# 0". For example, CALL('Netname',SPACE(15)) + '# 0'.

Returns

The name of the current network.

Scroll.bin

Allows an area on the screen to be scrolled up and down. The scroll area is not cleared first as under dBASE IV.

Parameter

A character string whose first 8 character denote the region to be scrolled. Each region value requires two characters. The order of the regions is: top, left, bottom, and right. The last three characters denote the number of lines to scroll.

Returns

False (.F).

Getdrive.bin

Returns the character drive designator.

Parameter

A 13 character string whose last character is a colon (:).

Returns

The default drive as a character string.

Tmpname.bin

Replaces the FoxBASE+ unique temporary filename.

Parameter

A 12 character string.

Returns

A unique filename in a character string.

Chapter 7: Supplemental Program Files

The following is a list of the supplemental program files included with Step IVward. These are needed for the conversion of several of functions that dBASE IV does not support. Some of the programs make CALLs to the binary files described in Chapter 6.

A_fields.prg

A user-defined function to replace Clipper's AFIELDS() function.

Parameters

1. The type of call.
2. A string buffer of from the previous type 1 call.

Returns

Return values for each type of call (indicated by the first parameter) are as follows:

1 - Buffer containing the first or next line of the .dbf structure listing
2 - Field name from the structure listing buffer
3 - Field type from the structure listing buffer
4 - Field width (numeric) from the structure listing buffer
5 - Field decimals (numeric) from the structure listing buffer

Asort.prg

A user-defined function to replace Clipper's ASORT() function.

Parameters

1. A numeric value indicating the type of the action. (1—posts values to a temporary array; 2—rebuilds the source array)
2. The value to post to the sort array. If the first parameter is 1, this is the source array element value. If the first parameter is 2, this is the offset to the temporary array.

Returns

The next sort value in the sequence. The objective of this value is dependent on the first parameter.

Bitset.prg

A user-defined function to replace Quicksilver's BITSET() function.

Parameters

1. A numeric expression to test.
2. The bit position within the first parameter to test.

Returns

True (.T.) if the specified bit is 1 in the binary representation of the first parameter.

C_file.prg

A routine to replace the Clipper FILE() function.

Parameter

Filename or file skeleton to search for.

Returns

True (.T.) if the file is found; otherwise, false (.F.) is returned.

Center.prg

A user-defined function to replace Quicksilver's CENTER() function.

Parameters

1. The character string to center.
2. The length of line in which to center the first parameter.

Returns

The first parameter centered within a string characters whose length is determined by the second parameter.

Dbedit.prg

A program to replace Clipper's DBEDIT() function.

Parameters

1. The top row for the BROWSE window.
2. The left column for the BROWSE window.
3. The bottom row for the BROWSE window.
4. The right column for the BROWSE window.

Returns

Nothing.

Descend.prg

A user-defined function to replace Clipper's DESCEND() function.

Parameter

An expression of any data type.

Returns

The complemented value of the expression. Memo fields are returned unchanged.

Empty.prg

User-defined function to replace the EMPTY() function.

Parameter

Expression to check.

Returns

True (.T.) if the expression result is empty; otherwise, false (.F.) is returned.

Fclose.prg

User-defined function to replace Clipper's FCLOSE() function.

Parameter

The file handle determined by FOPEN or FCREATE.

Returns

True (.T.) if no errors occurred; otherwise, false (.F.) is returned. The return value is stored in the memory variable f_error.

Fcount.prg

User-defined function to replace Clipper's FCOUNT() function.

Parameter

None.

Returns

The number of fields in current database file.

Fcreate.prg

User-defined function to replace Clipper's FCREATE() function.

Parameters

1. The name of the file to create.
2. The DOS file attribute for the new file.

Returns

Numeric value containing the file handle or −1 if an error occurred. The return value is stored in the memory variable f_error.

Ferror.prg

A user-defined function to replace Clipper's FERROR() function.

Parameter

None.

Returns

The DOS error number from the last file function.

Fopen.prg

A user-defined function to replace Clipper's FOPEN() function.

Parameters

1. The name of the file to open.
2. The mode to open the file.

Returns

A numeric value containing the file handle or −1 if an error occurred. The return value is stored in the memory variable f_error.

Fread.prg

A user-defined function to replace Clipper's FREAD() function.

Parameters

1. The file handle determined by FOPEN or FCREATE.
2. The number of bytes to read.

Returns

A character string with either the specified data or, a null string if an error occurred. The return value is stored in the memory variable f_error.

Freadstr.prg

A user-defined function to replace Clipper's FREADSTR() function.

Parameters

1. A file handle determined by FOPEN or FCREATE.
2. The number of bytes to read.

Returns

A character string containing the read data or, null string if an error occurred. The return value is stored in the memory variable f_error.

Fseek.prg

A user-defined function to replace Clipper's FSEEK() function.

Parameters

1. The file handle determined by FOPEN or FCREATE.
2. The number of bytes to move the file pointer; the value can be negative.
3. Method of moving the pointer.

Returns

The new position of the file pointer relative to the beginning of the file. The return value is stored in the memory variable f_error.

Fwrite.prg

A user-defined function to replace Clipper's FWRITE() function.

Parameters

1. The file handle determined by FOPEN or FCREATE.
2. A character string to write to the file.
3. The number of bytes to write to the file.

Returns

The number of bytes written or, 0 if an error occurred. The return value is stored in the memory variable f_error.

Header.prg

A user-defined function to replace Clipper's HEADER() function.

Parameter

None.

Returns

The length of the header area of the currently selected database file. (See RECSIZE() in the dBASE IV *Language Reference* manual for details on calculating the size of the .dbf file header.)

Htoi.prg

A user-defined function to replace Quicksilver's HTOI() function.

Parameter

A character string containing the hexadecimal representation of a number.

Returns

The decimal representation of the parameter as an integer.

Indexext.prg

A user-defined function to replace Clipper's INDEXEXT() function.

Parameter

None.

Returns

A character string whose value is the filename extension of the current index file.

Indexord.prg

A user-defined function to replace Clipper's INDEXORD().

Parameter

None.

Returns

The numeric position of the controlling index.

Itoh.prg

A user-defined function to replace Quicksilver's ITOH() function.

Parameters

1. A numeric expression to convert.
2. A numeric expression whose result determines if the return value is padded with leading zeroes.

Returns

A character string which is the hexadecimal representation of the first parameter.

Memoline.prg

A routine to replace the Clipper MEMOLINE() function.

Parameters

1. The memo field or character string to process.
2. The width of the text.
3. The line number.
4. The tab stop settings. (Not used.)
5. A logical to determine if wrapping occurs. (Not used.)

Returns

A line of text if available; otherwise, a null character string.

Memowrit.prg

A routine to replace the Clipper MEMOWRIT() function.

Parameters

1. The destination file.
2. The memo field or character string to process.

Returns

True (.T.) if successful; otherwise, false (.F.) is returned.

Mlcount.prg

A routine to replace the Clipper MLCOUNT() function.

Parameters

1. The memo field or character string to process.
2. The width of the text.
3. (Not used)
4. A logical that determines if word-wrapping occurs. (Not used)

Returns

The number of lines of wrapped text.

Mlpos.prg

A routine to replace the Clipper MLPOS() function.

Parameters

1. The memo field or character string to process.
2. The width of the text.
3. The number of lines to search.

Returns

The line number of the sought after text.

Proper.prg

A user-defined function to replace Quicksilver's PROPER() function.

Parameter

A character string to convert.

Returns

A character string of the parameter with an initial capital.

Rat.prg

A user-defined function to replace Clipper's RAT() function.

Parameters

1. The character string to search for.
2. The character string to be searched.

Returns

The starting position of the last instance of the first parameter within the second parameter.

Scroll.prg

A user-defined function to replace Clipper's SCROLL() function.

Parameters

1. Screen coordinates.
2. The number of lines to scroll. (Negative values are accepted.)

Returns

The screen is scrolled and False (.F) is returned.

Seconds.prg

A user-defined function to replace Clipper's SECONDS() function.

Parameter

None.

Returns

The number of seconds since midnight.

Selected.prg

A user-defined function to replace Clipper's SELECT() function.

Parameter

None.

Returns

The currently selected work area number.

Setesc.prg

A user-defined function to replace Clipper's SETCANCEL() function.

Parameters

An optional new value for SET ESCAPE.

Returns

The current SET ESCAPE setting (.T. = ON; .F. = OFF).

Setprc.prg

A user-defined function to replace Clipper's SETPRC() function.

Parameters

1. The new internal row position.
2. The new internal column position.

Returns

False (.F.). The dBASE IV system memory variables _prow and _pcol are reset. Note that in dBASE IV, the print head is not moved unless the changes occur in a PRINTJOB.

Sinkey.prg

A user-defined function to replace Quicksilver's SINKEY() function.

Parameters

None.

Returns

A one-byte or two-byte character string.

Strtran.prg

A user-defined function to replace Clipper's STRTRAN() function.

Parameters

1. The character string to search.
2. The character string to locate.
3. The replacement character string.
4. The number of the first occurrence to replace.
5. The number of occurrences to replace.

Returns

The updated character string.

Sys.prg

A user-defined function to replace the FoxBASE+ SYS() functions.

Parameters

See Chapter 4 for details.

Returns

See Chapter 4 for details.

Tone.prg

A user-defined function to replace Clipper's TONE() function.

Parameters

1. The frequency in cycles per second for the tone.
2. The duration in increments of 1/18 of a second.

Returns

The tone is sounded and false (.F.) is returned.

Word.prg

A user-defined function to replace Clipper's WORD() function.

Parameters

A numeric value.

Returns

The integer of the parameter.

Appendix A: Messages

Internal Error Messages

When one of these error messages occurs, there is a problem with the Step IVward system.

Unable to access overlay.

The Step.ovr file is not in the same directory as the Step.exe file.

Can't open disk file for print:

A DOS file opening error has occurred. Errors could be disk is full, duplicate filename, or insufficient rights to the directory.

Error writing output file. (Disk full or write protected?)

Erase some files on the disk or, remove the write protect tab, and then rerun Step IVward.

Stack full

This error occurs when there are too many programs or procedures for Step IVward to translate at one time. The limit is 255.

Messages Added to Program Files

Step IVward prefixes all messages written to program files with one of the following statements set of symbols:

*:: Denotes original commands that were commented out.

*|| Denotes a warning.

*!! Denotes an error.

After the three character note symbols, one of the following messages will be included:

*** Warning:

*** Error:

*** Blocking Error:

The messages that Step IVward inserts are as follows:

command/function not allowed in a dBASE IV FUNCTION

function returns different values under dBASE IV

function return value not available - constant inserted

LOOP or EXIT statement not within block

maximum string length is 254 under dBASE IV

program flow error due to illegal

RECNO(0) - you must SET NEAR ON prior to SEEK in dBASE IV

statement converted but, behavior different under dBASE IV

statement deleted — capability not supported under dBASE IV

statement deleted — not required under dBASE IV

statement deleted — requires programmer attention

*** Syntax error:

TO expected

Unexpected end of statement