

Update

Microsoft<sup>®</sup> FOXPRO<sup>®</sup>

*Relational Database Management System for MS-DOS<sup>®</sup>*

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation.

© 1989–1993 Microsoft Corporation. All rights reserved.

Microsoft, the Fox Head logo, FoxBASE+, FoxPro, MS, MS-DOS, and Multiplan are registered trademarks and Windows is a trademark of Microsoft Corporation in the United States of America and other countries.

Paradox is a registered trademark of Ansa Software, a Borland company.

Macintosh is a registered trademark of Apple Computer, Inc.

dBASE III PLUS, dBASE IV, and RapidFile are registered trademarks and Framework II is a trademark of Ashton-Tate Corporation.

Lotus, Symphony, and 1-2-3 are registered trademarks of Lotus Development Corporation.

UNIX is a registered trademark of UNIX Systems Laboratories.

## Contents

### FoxPro 2.5 New Features

Enhancements to the 32-Bit Product . . . . .	1-2
Provides 225 Work Areas . . . . .	1-2
Operates with DPMI-Compliant Memory Managers . . . . .	1-2
Support for Cross Platform Development . . . . .	1-3
New or Enhanced Commands and Functions . . . . .	1-4
New Generator Directives . . . . .	1-41
New System Memory Variables . . . . .	1-46

### Quick Start

Some Navigation Aids . . . . .	2-2
FoxPro Makes It Easy to Find Information . . . . .	2-3
Three Steps to RQBE . . . . .	2-5
Browsing a Database File . . . . .	2-6
Turning Data into Information . . . . .	2-7
Saved Queries Save You Time and Effort . . . . .	2-9
Changing Queries Is Easy . . . . .	2-10
Applications Can Make Tough Jobs Simple . . . . .	2-11
It's Easy to Get Just the Data You Want . . . . .	2-13
"Quick Reports" Really Are Quick and Easy . . . . .	2-15
Letting the Report Writer Work for You . . . . .	2-17
Even Multi-File Reports Are Easy . . . . .	2-19
Grouping and Subtotaling in RQBE . . . . .	2-21
One Last Pass with the Report Writer . . . . .	2-23
Data Editing and Entry—Your First Application . . . . .	2-25
Where Do You Go From Here? . . . . .	2-27

### Using Files From Other Platforms

Cross Platform Glossary . . . . .	3-3
Running MS-DOS Applications in Windows . . . . .	3-4
Approach 1: Running an MS-DOS Application As Is . . . . .	3-6
Approach 2: Transporting MS-DOS Applications . . . . .	3-12
Approach 3: Taking Full Advantage of Windows Features . . . . .	3-17

## **Using Files From Other Platforms (cont)**

Maintaining Cross Platform Files . . . . .	3-19
Maintaining Platform Screens . . . . .	3-19
Maintaining Cross Platform Reports . . . . .	3-23
Maintaining Cross Platform Labels . . . . .	3-24
Maintaining Cross Platform Menus . . . . .	3-24
Running Windows Applications in MS-DOS . . . . .	3-26
Transporter – How Does It Work? . . . . .	3-27
Transport Dialog . . . . .	3-27
Special Transporter Decisions . . . . .	3-28
Choosing a Development Platform . . . . .	3-29
Character Mode MS-DOS Platform . . . . .	3-29
Graphical Windows Platform . . . . .	3-29

## **Tables**

File Types and Extensions . . . . .	4-2
System Capacities . . . . .	4-4
Control Key Shortcuts . . . . .	4-6
Table Structures . . . . .	4-7
.PJX Table (Projects) . . . . .	4-8
.SCX Table (Screens) . . . . .	4-10
.SCX Table (Continued) . . . . .	4-12
.FRX Table (Reports) . . . . .	4-14
.MNX Table (Menus) . . . . .	4-18
.LBX Table (Labels) . . . . .	4-19
Table File Structure (.DBF) . . . . .	4-20
Memo and General File Structure (.FPT) . . . . .	4-22
Index File Structure (.IDX) . . . . .	4-23
Compact Index File Structure (.IDX) . . . . .	4-26
Compound Index File Structure (.CDX) . . . . .	4-29
FoxPro 2.0 Project File Structure (.PJX) . . . . .	4-30
FoxPro 2.0 Screen File Structure (.SCX) . . . . .	4-32
FoxPro 2.0 Report File Structure (.FRX) . . . . .	4-34
FoxPro 2.0 Label File Structure (.LBX) . . . . .	4-36
FoxPro 1.x Report File Structure (.FRX) . . . . .	4-37
FoxPro 1.x Label File Structure (.LBX) . . . . .	4-40
FoxBASE+ Memo File Structure (.DBT) . . . . .	4-41
FoxPro Macro File Format (.FKY) . . . . .	4-42



# 1 FoxPro 2.5 New Features

Microsoft® FoxPro® 2.5 for MS-DOS® is faster than FoxPro 2.0 and contains these new features and enhancements:

- Expanded support for the 32-bit product (see the Enhancements to the 32-Bit Product section)
- Support for the development of cross platform applications (see the Cross Platform Development section)
- Support for importing Microsoft Excel 3.0, Microsoft Excel 4.0, Paradox® 3.5 and Paradox 4.0 files (see the IMPORT and APPEND FROM commands in the New or Enhanced Commands and Functions section)
- An expanded record size that includes up to 65,000 bytes per record
- New or enhanced commands and functions (see the New or Enhanced Commands and Functions section)
- New generator directives (see the New Generator Directives section)
- New system memory variables (see the New System Memory Variables section)

This chapter describes these features in detail.

FoxPro 2.0 screen, report, label, menu and project files are not compatible with FoxPro 2.5 format. When you first open files of these types, FoxPro 2.5 converts them to FoxPro 2.5 format (with your permission). For more information about FoxPro 2.5 file formats, see the Tables chapter.

Library files created with the FoxPro 2.0 Library Construction Kit cannot be used in FoxPro 2.5. To use them with FoxPro 2.5, you must first update them. For details about updating your libraries, contact the vendor that provides them.

Additionally, programs compiled under FoxPro 2.0, such as .APPs and .FXPs, are not compatible with FoxPro 2.5. To run such programs in FoxPro 2.5, you must rebuild them. For more information about rebuilding FoxPro 2.0 applications, see the Using Files From Other Platforms chapter.

## Enhancements to the 32-Bit Product

---

The 32-bit products offers 225 work areas and operates with DOS Protected Mode Interface (DPMI) memory managers. The following sections describe these enhancements.

### **Provides 225 Work Areas**

FoxPro 2.5 features 225 work areas. With so many work areas, you can open many databases. This enhances the performance of applications that use many databases, because you can leave the databases open instead of closing and reopening them frequently.

### **Operates with DPMI-Compliant Memory Managers**

FoxPro 2.5 works with DPMI-compliant memory managers, such as Microsoft Windows™. If you use a DPMI memory manager, you can control the physical memory that FoxPro uses by specifying the MEMLIMIT option in your CONFIG.FP file.

The syntax for MEMLIMIT is:

```
MEMLIMIT = <percent>[, <minimum>, <maximum>]
```

where:

<percent> is the percent of available physical memory that you want FoxPro to use.

<minimum> is the minimum amount of memory required, expressed in kilobytes (K).

<maximum> is the maximum amount of memory required, expressed in kilobytes (K).

The minimum and maximum values are optional, but if you specify a maximum value you must also specify a minimum.

## **Support for Cross Platform Development**

FoxPro version 2.5 allows you to create and maintain applications that run on multiple platforms, such as Windows and MS-DOS. An application that can run on multiple platforms is a *cross platform* application.

With cross platform applications, you can run an application on multiple platforms and share data with full record locking and other multi-user capabilities. For information about cross platform application development, refer to the Using Files From Other Platforms chapter.

## **New or Enhanced Commands and Functions**

In version 2.5, the following commands and functions are new or enhanced:

- **APPEND FROM** — Appends files from Microsoft Excel 3.0, Microsoft Excel 4.0, Paradox 3.5 and 4.0 to FoxPro tables.
- **#DEFINE ... #UNDEF** — Creates and releases compile-time constants.
- **GETDIR()** — Displays the Select Directory dialog.
- **#IF ... #ENDIF** — Conditionally includes source code at compile time.
- **IMPORT** — Imports files from Microsoft Excel 3.0, Microsoft Excel 4.0, Paradox 3.5 and 4.0.
- **MOVE WINDOW** — Includes the **CENTER** clause that centers a window on the desktop or in a parent window.
- **SELECT - SQL** — Sends query results to a Browse window by default and includes the **NOWAIT**, **PREFERENCE**, and **TO SCREEN** clauses.
- **SET DEVELOPMENT** — Permits you to cancel program execution during a **READ** operation.
- **SHOW WINDOW** — Includes the **REFRESH** clause for updating memo editing windows.
- **SYS(20)** — Transforms German text.
- **USE** — Supports using 0 as a work area and includes the **SHARE** clause that opens a table/.DBF for shared use on a network.

These language changes are described in the following sections and in the online help file.

## APPEND FROM

---

<b>Purpose</b>	Adds records from another file to the end of the current table/.DBF.
<b>Syntax</b>	<b>APPEND FROM &lt;file&gt;   ?</b> [ <b>FIELDS &lt;field list&gt;</b> ] [ <b>FOR &lt;expl&gt;</b> ] [[ <b>TYPE</b> ] [ <b>DELIMITED</b> [ <b>WITH TAB</b>   <b>WITH &lt;delimiter&gt;</b>   <b>WITH BLANK</b> ]   <b>DIF</b>   <b>FW2</b>   <b>MOD</b>   <b>PDOX</b>   <b>RPD</b>   <b>SDF</b>   <b>SYLK</b>   <b>WK1</b>   <b>WK3</b>   <b>WKS</b>   <b>WR1</b>   <b>WRK</b>   <b>XLS</b> ]]
<b>See Also</b>	COPY FILE, COPY TO, IMPORT, EXPORT

**Description** The file you are appending from is assumed to be a FoxPro table with a .DBF extension. If the file you want to append from is a FoxPro table and doesn't have a .DBF extension, you must specify its extension. If the file is not a FoxPro table, you must specify the type of file you append from.

Before you can append from a table created in dBASE IV<sup>®</sup> that contains a memo field, you must first open the table in FoxPro with USE. You are prompted with "Convert MEMO file to FoxPro Format?" Choose **Yes**.

If you append from a FoxPro table, the table you append from can be open in another work area. You can also append from a table that isn't open but is available on disk and a shared table opened when SET EXCLUSIVE is OFF. When the table you append from contains records marked for deletion, the records are not marked for deletion after they are appended.

If you include the ? clause instead of including a table name, the Open dialog appears so you can choose a table to append from.

### **Clauses**     **<file>**

Specify the name of the file to append from with <file>. If you don't include a file name extension, the default extension .DBF is assumed.

### **FIELDS <field list>**

APPEND FROM supports an optional <field list>. Data is only appended to the fields specified in the field list.

## **FOR <expL>**

The entire source file is appended to the table unless you include the FOR clause. If the FOR clause is included, a new record is appended for each record in the file source for which <expL> evaluates to a logical true (.T.). Records are appended until the end of the file is reached.

## **TYPE**

If the file you are appending from isn't a FoxPro table, you must specify the file TYPE. Although you must specify the file type, you need not include the key word TYPE. You can append from a wide variety of different file types including DELIMITED ASCII text files in which you can specify a field delimiter.

If the file you are appending from doesn't have the usual default file extension for that type of file, the source file name must include the file's extension. For example, Microsoft Excel spreadsheets normally have an .XLS file name extension. If the spreadsheet you are appending from has an extension other than the expected .XLS, be sure to specify the extension.

When appending from a spreadsheet, the data in the spreadsheet must be stored in a row major order rather than a column major order. This allows the appended spreadsheet data to match the table structure.

## **DELIMITED [WITH TAB | WITH <delimiter> | WITH BLANK]**

A DELIMITED file is an ASCII text file in which each record ends with a carriage return and line feed. Field contents are by default assumed to be separated from each other by commas, and character field values to be additionally delimited by double quotation marks. For example:

```
''Smith'', 99999999, ''TELEPHONE''
```

The DELIMITED WITH TAB option can be used to specify files which contain fields separated from each other by tabs rather than commas. The DELIMITED WITH <delimiter> option can be used to indicate that character fields are delimited by a character other than the quotation mark. The DELIMITED WITH BLANK option can be used to specify files which contain fields separated by spaces instead of commas. The file extension is assumed to be .TXT for all delimited files.

You can import dates from delimited files if the dates are in proper date format. The date format defaults to 'mm/dd/yy'. Including the century portion of a date is optional. FoxPro will import a date that includes the century. If the century isn't included in a date (for example '12/25/92'), the Twentieth century is assumed. Date delimiters can be any non-numeric character except the delimiter that separates the fields in the delimited file.

Dates in other formats can be imported if their format matches a date format available in SET DATE. To import dates that are not in the default format, issue SET DATE with the proper date format before using APPEND FROM. To test if a date format can be successfully imported, use it with CTOD( ). If the date is acceptable to CTOD( ), the date will import properly.

### **DIF**

In a DIF (VisiCalc Data Interchange Format) file, vectors (columns) become fields and tuples (rows) become records. DIF file names are assumed to have a .DIF extension.

### **FW2**

FW2 files are created by Framework II™. FW2 file names are assumed to have a .FW2 extension.

### **MOD**

MOD files are created by Microsoft Multiplan® version 4.01. MOD files are assumed to have a .MOD extension.

### **PDOX**

Database files in Paradox versions 3.5 and 4.0 can be appended to a FoxPro table by including the PDOX option. Paradox file names are assumed to have a .DB extension.

### **RPD**

RPD files are created by RapidFile® version 1.2. RPD file names are assumed to have an RPD extension.

### **SDF**

An SDF (System Data Format) file is an ASCII text file in which records have a fixed length and end with a carriage return and a line feed. Fields are not delimited. The file name extension is assumed to be .TXT for SDF files.

## **SYLK**

An SYLK file is in a Symbolic Link interchange format (used in Microsoft Multiplan) in which columns become fields and rows become records. SYLK file names have no extension by default.

## **WK1**

Data from a Lotus<sup>®</sup> 1-2-3<sup>®</sup> spreadsheet can be appended to a FoxPro table with this option. Each column from the spreadsheet becomes a field in the table; each spreadsheet row becomes a record in the table. A .WK1 file name extension is assigned to a spreadsheet created in Lotus 1-2-3 revision 2.x.

## **WK3**

Data from a Lotus 1-2-3 spreadsheet. Each column from the spreadsheet becomes a field in the table; each spreadsheet row becomes a record in the table. A .WK3 file name extension is assigned to a spreadsheet created in Lotus 1-2-3 revision 3.x.

## **WKS**

Data from a Lotus 1-2-3 spreadsheet. Each column from the spreadsheet becomes a field in the table and each spreadsheet row becomes a record in the table. A .WKS file name extension is assigned to a spreadsheet created in Lotus 1-2-3 revision 1-A.

## **WR1**

Data from a Lotus Symphony<sup>®</sup> spreadsheet. Each column from the spreadsheet becomes a field in the table and each spreadsheet row becomes a record in the table. A .WR1 file name extension is assigned to a spreadsheet created in Symphony versions 1.1 or 1.2.

## **WRK**

Data from a Lotus Symphony spreadsheet. Each column from the spreadsheet becomes a field in the table and each spreadsheet row becomes a record in the table. A .WRK file name extension is assigned to a spreadsheet created in Symphony version 1.0.

## **XLS**

The data in Microsoft Excel versions 2.0, 3.0 and 4.0 spreadsheets can be appended to a FoxPro table by including the XLS option. Each column from the spreadsheet becomes a field in the table and each spreadsheet row becomes a record in the table. Spreadsheet files created in Microsoft Excel are given an .XLS file name extension.



**Example**

In this example, CUSTOMER.DBF is opened, its structure is copied to a table called BACKUP and BACKUP is opened. FoxPro then appends all records from CUSTOMER.DBF from the state of Ohio (OH). These records are then copied to a new delimited file called TEMP.TXT.

```
CLOSE DATABASES
USE customer
COPY STRUCTURE TO backup
USE backup
APPEND FROM customer FOR state = 'OH'
COPY TO temp TYPE DELIMITED
MODIFY FILE temp.txt
USE
DELETE FILE backup.dbf
DELETE FILE temp.txt
```

## #DEFINE ... #UNDEF

---

<b>Purpose</b>	Creates and releases compile-time constants
<b>Syntax</b>	<b>#DEFINE &lt;constant name&gt; &lt;expr&gt;</b>  <b>#UNDEF &lt;constant name&gt;</b>
<b>See Also</b>	COMPILE, #IF ... #ENDIF

---

**Description** The #DEFINE and #UNDEF preprocessor directives are used to create compile-time constants in programs. By creating constants with #DEFINE instead of using system memory variables, you can reduce memory consumption, increase performance and simplify programs.

To create a constant with #DEFINE, specify the constant's name with <constant name> and its value with <expr>. When the program is compiled, text substitution is performed and the constant value expression is substituted for the constant name wherever it appears in the program. You can stop the substitution for the constant by issuing #UNDEF.

Substitution occurs only in program lines that follow the #DEFINE statement that creates the constant and that precede the #UNDEF statement for that constant. The constant is available only to the program that creates the constant.

### Clauses <constant name>

To create a compile-time constant with #DEFINE, specify its name with <constant name>. The constant name must be a legitimate FoxPro name that begins with a letter or an underscore and consists of up to 10 letters, digits or underscores. To improve program readability and simplify debugging, use a standard naming convention for your constant names.

Don't use FoxPro key words for constant names.

To stop text substitution for a constant created with #DEFINE, issue #UNDEF <constant name>.

**<expr>**

Specify the value of the compile-time constant with <expr>. <expr> can be a name or an expression that evaluates to a character, numeric, date or logical value.

**Example**

The following program creates a compile-time constant named MAXITEMS. This constant is used in a FOR ... NEXT loop to display the numbers 1 through 10.

```
#DEFINE maxitems 10

FOR i = 1 TO maxitems
  ? i
NEXT
```

## GETDIR( )

---

<b>Purpose</b>	Displays the Select Directory dialog
<b>Syntax</b>	<b>GETDIR([&lt;expC1&gt; [, &lt;expC2&gt;]])</b>
<b>Parameters</b>	 <expC1>    The directory initially selected <expC2>    The prompt displayed at the top of the dialog
<b>Returns</b>	Character
<b>See Also</b>	GETEXPR

---

**Description** GETDIR( ) displays the Select Directory dialog from which a directory can be chosen. GETDIR( ) returns the name of the directory you choose as a character string.

If you do not choose a directory (you choose **Cancel** or press Escape), GETDIR( ) returns the null string.

**Parameters** **<expC1>**

Use <expC1> to specify the directory that is initially displayed in the Select Directory dialog. When <expC1> is not specified, the Select Directory dialog opens with the default directory displayed.

**<expC2>**

The prompt for the directory list in the Select Directory dialog is specified with <expC2>.

## #IF ... #ENDIF

---

<b>Purpose</b>	Conditionally includes source code at compile time
<b>Syntax</b>	<pre>#IF &lt;expN1&gt;   #IF &lt;expL1&gt;     &lt;statements&gt; [#ELIF &lt;expN2&gt;   #ELIF &lt;expL2&gt;     &lt;statements&gt; ... #ELIF &lt;expNn&gt;   #ELIF &lt;expLn&gt;     &lt;statements&gt;] [#ELSE     &lt;statements&gt;] #ENDIF</pre>
<b>See Also</b>	COMPILE, #DEFINE ... #UNDEF

<b>Description</b>	<p>The #IF ... #ENDIF preprocessor directives let you conditionally include source code in a compiled program. #IF ... #ENDIF can improve the readability of source code, reduce compiled program size and improve performance in some cases.</p> <p>When the #IF ... #ENDIF structure is compiled, successive logical or numeric expressions within the structure are evaluated; the evaluation results determine which set of FoxPro statements (if any) are included in the compiled code.</p>
<b>Clauses</b>	<p><b>#IF &lt;expN1&gt;   #IF &lt;expL1&gt;</b> <b>&lt;statements&gt;</b></p> <p>The numeric expression &lt;expN1&gt; or logical expression &lt;expL1&gt; is evaluated. If the expression is numeric and non-zero or the expression is logical and true (.T.), the statements immediately following #IF are included in the compiled code. The #IF ... #ENDIF structure is exited, and the first program line following #ENDIF is then compiled.</p> <p>If the expression following #IF is numeric and 0 or logical and false (.F.), the statements immediately following #IF are not included in the compiled code. In this case, any following #ELIF directives are evaluated.</p>

```
[#ELIF <expN2> | #ELIF <expL2>  
<statements>
```

...

```
#ELIF <expNn> | #ELIF<expLn>  
<statements>]
```

If <expN1> is 0 or <expL1> is false, the #ELIF directives are then evaluated. The first #ELIF expression <expN2> or <expL2>, if present, is evaluated. If <expN2> is non-zero or <expL2> is true, the statements immediately following #ELIF are included in the compiled code. The #IF ... #ENDIF structure is exited, and the first program line following #ENDIF is then compiled.

If <expN2> is 0 or <expL2> is false, the statements following #ELIF are not included in the compiled code, and the next #ELIF directive is evaluated.

```
[#ELSE  
<statements>]
```

If no #ELIF directives are included or if those that are included evaluate to 0 or false, the presence or absence of #ELSE determines whether any additional statements are included in the compiled code:

- If #ELSE is included, the statements following #ELSE are included in the compiled code.
- If #ELSE isn't included, none of the program statements between #IF and #ENDIF is included in the compiled code. The #IF ... #ENDIF structure is exited, and compilation continues on the first program line following #ENDIF.

## Example

In the following example, the #IF ... #ENDIF structure determines which version of FoxPro compiles the program and then displays the appropriate message.

```
#IF 'WINDOWS' $ UPPER(VERSION( ))  
  ? 'This is FoxPro for Windows'  
#ELIF 'MAC' $ UPPER(VERSION( ))  
  ? 'This is FoxPro for Mac'  
#ELIF 'UNIX' $ UPPER(VERSION( ))  
  ? 'This is FoxPro for UNIX'  
#ELSE  
  ? 'This is FoxPro for DOS'  
#ENDIF
```

# IMPORT

---

<b>Purpose</b>	Imports data in an external file format to create a new FoxPro table/.DBF
<b>Syntax</b>	<b>IMPORT FROM &lt;file&gt;</b> <b>[TYPE] FW2   MOD   PDOX   RPD</b> <b>  WK1   WK3   WKS   WR1   WRK   XLS</b>
<b>See Also</b>	APPEND FROM, COPY TO, EXPORT

**Description** Most software packages store their data in a file format that cannot be opened directly in FoxPro. IMPORT creates a new FoxPro table/.DBF from data stored in file formats that FoxPro cannot read.

A new table/.DBF is created with the same name as the file the data is imported from. A .DBF extension is assigned to the newly created table/.DBF.

**Clauses** **<file>**  
**<file>** is the name of the file to import data from. If you don't include an extension with the file name, the default extension for the specified file type is assumed.

## TYPE

The key word TYPE is optional, but you must include one of the following file types described below.

## FW2

Include FW2 to import FW2 files, created by Framework II.

## MOD

Include MOD to import MOD files, created by Microsoft Multiplan version 4.01.

## PDOX

Include PDOX to import Paradox files. Database files in Paradox versions 3.5 and 4.0 by Borland can be imported by including the PDOX option.

## RPD

Include RPD to import RPD files, created by RapidFile.

**WK1 | WK3 | WKS**

Include WK1 to import data from a Lotus 1-2-3 spreadsheet. Columns from the spreadsheet become fields in the table/.DBF, and the spreadsheet rows become records in the table/.DBF. A WK1 extension is assigned to spreadsheets created in Lotus 1-2-3 revision 2.x; a WK3 extension is assigned to spreadsheets created in Lotus 1-2-3 revision 3.x; and a .WKS extension is assigned to spreadsheets created in Lotus 1-2-3, revision 1-A.

**WR1 | WRK**

Include WR1 to import data from a Lotus Symphony spreadsheet. Columns from the spreadsheet become fields in the table/.DBF, and the spreadsheet rows become records in the table/.DBF. A WR1 extension is assigned to spreadsheets created in Symphony version 1.10, and a .WRK extension is assigned to spreadsheets created in Symphony version 1.01.

**XLS**

Include XLS to import data from Microsoft Excel spreadsheets versions 2.0, 3.0 and 4.0. Columns from the spreadsheet become fields in the table/.DBF, and the spreadsheet rows become records in the table/.DBF. Spreadsheet files created in Microsoft Excel have an .XLS extension.



## MOVE WINDOW

---

<b>Purpose</b>	Moves a window to a specified screen location
<b>Syntax</b>	<b>MOVE WINDOW</b> <window name> <b>TO</b> <row, column>   <b>BY</b> <expN1>, <expN2>   <b>[CENTER]</b>
<b>See Also</b>	ACTIVATE WINDOW

---

**Description** Use MOVE WINDOW to move a user-defined or system window to a specified location on the desktop in FoxPro for MS-DOS, the main FoxPro window in FoxPro for Windows or a user-defined window. The window can be moved to a specific position or relative to its current position. If a window is defined, it can be moved; it doesn't have to be active or visible.

**Clauses** <window name>

Specify the name of the window to move with <window name>.

**TO <row, column>**

Include this clause to specify the row and column coordinates to move a window to on the desktop, the main FoxPro window or a user-defined window.

**BY <expN1>, <expN2>**

Include this clause to move a window to a location relative to its current position. The first numeric expression <expN1> specifies the number of rows to move the window (down if <expN1> is positive, up if negative). The second numeric expression <expN2> specifies the number of columns to move the window (to the right if <expN2> is positive, to the left if negative).

**CENTER**

You can center a window on the desktop, the main FoxPro window, or in a parent window by including CENTER.

**Example**      In this example, after the window menter is defined and activated, the window is moved.

```
DEFINE WINDOW menter FROM 10,4 TO 15,54 SYSTEM ;  
    TITLE "Nomadic Window"  
ACTIVATE WINDOW menter  
WAIT WINDOW 'Press any key to move window'  
MOVE WINDOW menter TO 20,15  
WAIT WINDOW 'Press any key to release window'  
RELEASE WINDOW menter
```

## SELECT — SQL

---

<b>Purpose</b>	Retrieves data from one or more table/.DBF
<b>Syntax</b>	<pre>SELECT [ALL   DISTINCT]       [&lt;alias&gt;.]&lt;select_item&gt; [AS &lt;column_name&gt;]       [, [&lt;alias&gt;.]&lt;select_item&gt; [AS &lt;column_name&gt;] ...] FROM &lt;table&gt; [&lt;local_alias&gt;] [, &lt;table&gt; [&lt;local_alias&gt;] ...] [[INTO &lt;destination&gt;]     [TO FILE &lt;file&gt; [ADDITIVE]     TO PRINTER [PROMPT]     TO SCREEN]] [PREFERENCE &lt;name&gt;] [NOCONSOLE] [PLAIN] [NOWAIT] [WHERE &lt;joincondition&gt; [AND &lt;joincondition&gt; ...]   [AND   OR &lt;filtercondition&gt; [AND   OR &lt;filtercondition&gt; ...]]] [GROUP BY &lt;groupcolumn&gt; [, &lt;groupcolumn&gt; ...]] [HAVING &lt;filtercondition&gt;] [UNION [ALL] &lt;SELECT command&gt;] [ORDER BY &lt;order_item&gt; [ASC   DESC]   [, &lt;order_item&gt; [ASC   DESC]...]]</pre>
<b>See Also</b>	CREATE QUERY, CREATE TABLE — SQL, INSERT — SQL, MODIFY QUERY, SET ANSI, SET PATH, _TALLY

**Description** SELECT is a SQL command you use to retrieve data from one or more table/.DBF. When you use SELECT to pose a query, FoxPro interprets the query and retrieves the specified data from the tables/.DBFs. SELECT is built into FoxPro like any other FoxPro command. You can create a SELECT query:

- In the Command window
- In a FoxPro program (like any other FoxPro command)
- In the RQBE window

When you issue SET TALK ON and execute SELECT, FoxPro displays the query time and the number of records in the results. \_TALLY stores the number of records in the query results.

**Clauses** [ALL | DISTINCT]

The SELECT clause specifies the fields, constants and expressions that will be displayed in the query results.

By default, all of the rows (ALL) in the query results are displayed. Include DISTINCT to exclude duplicates of any rows from the query results.

You can use DISTINCT only once per SELECT clause.

**[<alias>.<select\_item> [AS <column\_name>]  
[, [<alias>.<select\_item> [AS <column\_name>] ...]**

<select\_item> can be one of the following:

- The name of a field from a table/.DBF in the FROM clause
- A constant specifying that the same constant value is to appear in every row of the query results
- An expression that can be the name of a user-defined function (UDF)

Each item you specify with <select\_item> generates one column of the query results. If more than one item has the same name, be sure to qualify the matching names by including the table/.DBF alias and a period before the item name.

Although using UDFs in the SELECT clause has obvious benefits, you should also consider the following restrictions:

1. The speed of operations performed with SELECT may be limited by the speed at which such UDFs are executed. High-volume manipulations involving UDFs may better be accomplished by using API and UDFs written in C or assembly language.
2. You can assume *nothing* about the FoxPro input/output (I/O) or table/.DBF environment in UDFs invoked from SELECT. In general, you don't know which work area is selected, the name of the current table/.DBF or even the names of the fields being processed. The value of these variables depend on precisely where in the optimization process the UDF is invoked.
3. It isn't safe to change the FoxPro I/O or table/.DBF environment in UDFs invoked from SELECT. In general, the results are unpredictable.
4. The *only* reliable way to pass values to UDFs invoked from SELECT is by the argument list passed to the function when it is invoked.
5. If you experiment and discover a supposedly forbidden manipulation that works correctly in a certain version of FoxPro, there is no guarantee that it will continue to work in later versions.

Apart from these restrictions, feel free to use UDFs. However, don't forget the effect of using SELECT on performance.

The following field functions are available for use with a select item that is a field or an expression involving a field:

- AVG(<select\_item>) — Averages a column of numeric data.
- COUNT(<select\_item>) — Counts the number of select items in a column. COUNT(\*) counts the number of rows in the query output.

- MIN(<select\_item>) — Determines the smallest value of <select\_item> in a column.
- MAX(<select\_item>) — Determines the largest value of <select\_item> in a column.
- SUM(<select\_item>) — Totals a column of numeric data.

You cannot nest field functions.

The optional AS <column\_name> clause specifies the heading for a column in the query output. This is useful when <select\_item> is an expression or contains a field function and you want to give the column a meaningful name. <column\_name> can be an expression but cannot contain characters (for example, spaces) that aren't permitted in table/.DBF field names.

### **FROM <table> [<local\_alias>] [, <table> [<local\_alias>] ... ]**

The FROM clause lists the tables/.DBFs that contain the data to be retrieved by the query. If no table/.DBF is open in a work area in the current directory or the FoxPro path, the Open dialog appears so you can specify the file location.

If a table/.DBF in the query isn't open in a work area, it is opened and remains open in a work area once the query is complete.

<local\_alias> is a temporary name for a table/.DBF specified with <table>. If you specify a local alias, you must use the local alias in place of the table/.DBF name throughout SELECT. The local alias doesn't affect the FoxPro environment.



If several tables have fields with the same name, which must be qualified, you may want to use a short local alias.

### **INTO <destination>**

The INTO clause determines where the query results are stored. If you include the INTO clause, no output display is produced. This means that if you include an INTO clause and a TO clause in the same query, the TO clause is ignored. If you don't include the INTO clause, query results are displayed in a Browse window. Query results can also be directed to the printer or a file with the TO clause.

<destination> can be one of the following:

- **ARRAY <array>** — Stores query results in a memory variable array named <array>. The array isn't created if the query selects 0 records.
- **CURSOR <cursor>** — Stores query results in a cursor named <cursor>. If you specify the name of an open table/.DBF, FoxPro closes the table/.DBF and creates a cursor by that name without warning if SET SAFETY is OFF. After SELECT is executed, the temporary cursor remains open and is active but is read-only. Once you close this temporary cursor, it is deleted. Cursors can exist as a temporary file on the SORTWORK drive.
- **DBF <table> | TABLE <table>** — Stores query results in a table/.DBF named <table>. If you specify a table/.DBF that is already open, FoxPro closes the table/.DBF and reopens it without warning if SET SAFETY is OFF. If you don't specify an extension, the table/.DBF is given a .DBF extension. The table/.DBF remains open and active after SELECT is executed.

**TO FILE <file> [ADDITIVE]**

**| TO PRINTER [PROMPT]**

**| TO SCREEN**

If you include a TO clause but not an INTO clause, you can direct query results to an ASCII text file named <file> or to the printer in addition to the desktop or main FoxPro window.

If query results are directed to a text file, including ADDITIVE appends output to any existing contents of the file. If you don't include a TO clause or an INTO clause, the query results appear on the screen by default unless you specify NOCONSOLE.

In FoxPro for Windows, if query results are directed to the printer, you can include the optional PROMPT clause to display a dialog before printing starts. Place the PROMPT key word immediately after TO PRINTER. PROMPT is ignored in FoxPro for MS-DOS.

In this dialog you can adjust printer settings. You can specify the number of copies and page numbers to print and direct print output to a file. The Windows Printer Setup dialog can also be opened to adjust additional printer settings. The printer settings you can adjust depend on the currently installed printer driver.

By default, query results are sent to a Browse window. Include **TO SCREEN** to direct query results to the desktop (FoxPro for MS-DOS), the main FoxPro window (FoxPro for Windows) or an active user-defined window.

When query results are output, columns are named according to the following rules:

- If a select item is a field with a unique name, the output column name is the field's name.
- If more than one select item has the same name (for example, **CUST.ZIP** and **STATE.ZIP**) output columns are named **<extension>\_A** and **<extension>\_B** (**ZIP\_A** and **ZIP\_B**). For a select item with a 10-character name, the name is truncated to add the underscore and letter.
- If a select item is an expression, its output column is named **EXP\_A**. Any other expressions are named **EXP\_B**, **EXP\_C**, and so on.
- If a select item contains a field function such as **COUNT( )**, the output column is named **CNT\_A**. If another select item contains **SUM( )**, its output column is named **SUM\_B**.

### **PREFERENCE <name>**

If query results are sent to a Browse window, you can include **PREFERENCE** to save the Browse window's attributes and options for later use. **PREFERENCE** saves the Browse window's attributes indefinitely in the **FOXUSER** resource file. Preferences can be retrieved at any time. Issuing **SELECT** with a **PREFERENCE <name>** for the first time creates the preference. Issuing **SELECT** later with the same preference name restores the Browse window to that preference state. When the Browse window is closed, the preference is updated.

If you exit a Browse window by pressing **Ctrl+Q**, any Browse window changes are not saved to the resource file.

### **NOCONSOLE**

Including **NOCONSOLE** prevents the query results from being displayed on the desktop or main FoxPro window. **NOCONSOLE** can be used whether or not a **TO** clause is present. If an **INTO** clause is included, **NOCONSOLE** is ignored.



## **PLAIN**

Including PLAIN prevents column headings from appearing in the query output displayed. PLAIN can be used whether or not a TO clause is present. If an INTO clause is included, PLAIN is ignored.

## **NOWAIT**

When query results are directed to a Browse window in a program, including NOWAIT continues program execution after the Browse window is opened. The program doesn't wait for the Browse window to be closed, but continues execution on the program line immediately following the SELECT statement.

When TO SCREEN is included to direct output to the desktop (FoxPro for Windows), the main FoxPro window (FoxPro for Windows) or a user-defined window, output pauses when the desktop or main FoxPro window is full of query results. Press a key to see the next set of query results. If NOWAIT is included, the query results are scrolled off the desktop, the main FoxPro window or the user-defined window without pausing for a key press. NOWAIT is ignored if included with the INTO clause.

## **WHERE <joincondition> [AND <joincondition> ...]**

**[AND | OR <filtercondition> [AND | OR <filtercondition> ... ]]**

WHERE is required to retrieve data from multiple tables. It tells FoxPro to include only certain records in the query results.

<joincondition> specifies fields that link the tables in the FROM clause. If you include more than one table/.DBF in a query, you should specify a join condition for every table/.DBF after the first.

If you include two table in a query and don't specify a join condition, every record in the first table/.DBF is joined with every record in the second table/.DBF as long as the filter conditions are met. Such a query can produce lengthy results.

Be careful when using functions (DELETED( ), RECNO( ), EOF( ), FOUND( ), RECCOUNT( ), and so on) that support an optional alias or work area in join conditions. Including an alias or work area in these functions may yield unexpected results. SELECT doesn't use your work areas; it performs the equivalent of USE ... AGAIN. Single-table queries that use these functions without an optional alias or work area will return proper results. However, multiple-table queries that use these functions — even without an optional alias or work area — may return unexpected results.

Use caution when joining tables with empty fields because FoxPro matches empty fields. For example, if you join on CUSTOMER.ZIP and INVOICE.ZIP, and CUSTOMER contains 100 empty zip codes and INVOICE contains 400 empty zip codes, the query output contains 40,000 extra records resulting from the empty fields. Use the EMPTY( ) function to eliminate empty records from the query output.

Multiple join conditions must be connected with the AND operator. Each join condition has the following form:

<field1> <comparison> <field2>

where <field1> is the name of a field from one table, <field2> is the name of a field from another table, and <comparison> is one of the following operators:

<comparison>
=
< >, !=, #
= =
>
> =
<
< =

When you use the = operator with strings, it acts differently depending on the setting of SET ANSI. When SET ANSI is OFF, FoxPro treats string comparisons in the manner with which Xbase users are familiar. When SET ANSI is ON, string comparisons follow ANSI standards.

<filtercondition> specifies the criteria that records must meet to be included in the query results. A query can include as many filter conditions as desired, connected with the AND or OR operator. You can also use the NOT operator to reverse the value of a logical expression or use EMPTY( ) to check for an empty field. <filtercondition> can take one of these forms:

**Form:** <field1> <comparison> <field2>

**Example:** customer.cust\_id = payments.cust\_id

**Form:** <field> <comparison> <expression>

**Example:** payments.amount >= 1000

**Form:** <field> <comparison> ALL (<subquery>)

**Example:** taxrate < ALL ;

(SELECT taxrate FROM customer WHERE state = "WA")

When the filter condition includes ALL, the field must meet the comparison condition for all values generated by the subquery before its record is included in the query results.

**Form:** <field> <comparison> ANY | SOME (<subquery>)

**Example:** taxrate < ANY ;

(SELECT taxrate FROM customer WHERE state = "WA")

When the filter condition includes ANY or SOME, the field must meet the comparison condition for at least one of the values generated by the subquery.

**Form:** <field> [NOT] BETWEEN <start\_range> AND <end\_range>

**Example:** customer.taxrate BETWEEN 5.50 AND 6.00

This example checks to see if the values in the field are within a specified range of values.

**Form:** [NOT] EXISTS (<subquery>)

**Example:** EXISTS ;

(SELECT \* FROM invoices WHERE customer.zip = invoices.zip)

This example checks to see if at least one row meets the criteria in the subquery. When the filter condition includes EXISTS, the filter condition evaluates to true unless the subquery evaluates to the empty set.

**Form:** <field> [NOT] IN <value\_set>

**Example:** customer.zip NOT IN ("98052", "98072", "98034")

When the filter condition includes IN, the field must contain one of the values before its record is included in the query results.

**Form:** <field> [NOT] IN (<subquery>)

**Example:** customer.cust\_id IN ;

(SELECT tranfile.cust\_id FROM tranfile WHERE tranfile.item="FoxPro")

Here, the field must contain one of the values returned by the subquery before its record is included in the query results.

**Form:** <field> [NOT] LIKE <expC>

**Example:** customer.state NOT LIKE "WA"

This filter condition searches for each field that matches <expC>. You can use the wildcard characters % (percent sign) and \_ (underscore) as part of <expC>. The underscore represents a single unknown character in the string.

A subquery is SELECT within a SELECT and *must* be enclosed in parentheses. You can have multiple subqueries at the same level (not nested) in the WHERE clause. Subqueries can contain multiple join conditions.

**GROUP BY <groupcolumn> [, <groupcolumn> ... ]**

Including the GROUP BY clause groups rows in the query based on values in one or more columns. <groupcolumn> can be the name of a regular table/.DBF field, or a field that includes an SQL field function, or a numeric expression indicating the location of the column in the result table/.DBF (the leftmost column number is 1).

**HAVING <filtercondition>**

If you include the HAVING clause, FoxPro will include groups in the query results as long as they meet the filter condition specified with <filtercondition>. <filtercondition> cannot contain a subquery.

HAVING <filtercondition> should be used with GROUP BY to specify the criteria that groups must meet to be included in the query results. HAVING can include as many filter conditions as desired, connected with the AND or OR operator. You can also use NOT to reverse the value of a logical expression.

A HAVING clause without a GROUP BY clause acts like a WHERE clause. You can use local aliases and field functions in the HAVING clause. Use a WHERE clause for faster performance if your HAVING clause contains no field functions.

### **[UNION [ALL] <SELECT command>]**

Include the UNION clause to combine the final results of one SELECT with the final results of another SELECT. By default, UNION checks the combined results and eliminates duplicate rows. Use the optional ALL key word to prevent UNION from eliminating duplicate rows from the combined results.

You can use parentheses to combine multiple UNION clauses.

UNION clauses follow these rules:

- You cannot use UNION to combine subqueries.
- Both SELECTs must output the same number of columns.
- Each column in the query results of one SELECT must have the same data type and width as the corresponding column in the other SELECT.
- Only the final SELECT can have an ORDER BY clause, which must refer to output columns by number. If an ORDER BY clause is included, it affects the entire result.

### **ORDER BY <order\_item> [ASC | DESC] [, <order\_item> [ASC | DESC] ... ]**

ORDER BY sorts the query results based on the data in one or more columns. Each order item must correspond to a column in the query results and can be one of the following:

- A field in a FROM table that is also a select item in the main SELECT clause (not in a subquery).
- A numeric expression indicating the location of the column in the result table. (The leftmost column is number 1.)

You can specify the optional DESC key word if you want your query results to appear in descending order according to the order item or items. ASC specifies an ascending order for query results and is the default for ORDER BY.

Query results appear unordered if you don't specify an order with ORDER BY.

**Examples** This example displays the names of all companies in the CUSTOMER.DBF (one field from one table).

```
SELECT customer.company ;  
FROM customer
```

This example displays the contents of three fields from two tables and joins the two tables based on the CUST\_ID field. It uses local aliases for both tables.

```
SELECT a.company, b.idate, b.itotal ;  
FROM customer a, invoices b ;  
WHERE a.cno = b.cno
```

This example displays only records with unique data in the specified fields.

```
SELECT DISTINCT a.company, b.idate, b.itotal ;  
FROM customer a, invoices b ;  
WHERE a.cno = b.cno
```

This example displays STATE, ZIP and COMPANY fields in ascending order.

```
SELECT state, zip, company, cno;  
FROM customer ;  
ORDER BY state, zip, company
```

This example stores the contents of fields from two tables in a third table.

```
SELECT a.company, b.idate, b.itotal ;  
FROM customer a, invoices b ;  
WHERE a.cno = b.cno ;  
INTO TABLE custinv.dbf  
BROWSE  
CLOSE DATA
```

This example displays only records with an invoice date earlier than 05/08/90.

```
SELECT a.company, b.idate, b.itotal ;  
FROM customer a, invoices b ;  
WHERE a.cno = b.cno ;  
AND b.idate {05/08/90}
```

This example displays the names of all companies from CUSTOMER.DBF with a zip code that matches a zip code in the INVOICES table/.DBF.

```
SELECT company FROM customer a WHERE ;  
      EXISTS (SELECT * FROM offices b WHERE a.zip = b.zip)
```

This example displays all records from CUSTOMER.DBF having a company name that begins with an upper-case C and is an unknown length.

```
SELECT * FROM customer a WHERE a.company LIKE "C%"
```

This example displays all records from CUSTOMER.DBF having a state name that begins with an upper-case N and is followed by one unknown character.

```
SELECT * FROM customer a WHERE a.state LIKE "N_"
```

This example displays the names of all cities in CUSTOMER.DBF in upper-case and names the output column CityList.

```
SELECT UPPER(city) AS CityList FROM customer
```

# SET DEVELOPMENT

---

Purpose	Causes FoxPro to compare the creation date and time of a program with those of its compiled object file when the program is run
Syntax	SET DEVELOPMENT ON   OFF
See Also	COMPILE, MODIFY COMMAND

---

**Description** SET DEVELOPMENT compares the creation date and time of a source program with those of its corresponding compiled object file. FoxPro recompiles the source program before it executes if SET DEVELOPMENT is ON and the source program is more current than its compiled object program. This ensures that the most current version of a program is executed.

The source and compiled versions of the program aren't compared if SET DEVELOPMENT is OFF. You may not always be executing the most current version of a program if SET DEVELOPMENT is OFF.

The most current version of a program modified with the FoxPro editor invoked with MODIFY COMMAND is always executed, regardless of the DEVELOPMENT setting.

SET DEVELOPMENT needs to be ON only when programs are modified outside of FoxPro. Using an external editor (for example, a TSR editor) may require you to issue CLEAR PROGRAM before you execute the modified program. See CLEAR PROGRAM in this chapter for additional information. Use SET DEVELOPMENT OFF for optimum performance.

When SET DEVELOPMENT is ON, program execution can be canceled during a READ. The **Cancel** option on the **Program** menu is enabled when SET DEVELOPMENT is ON and a READ is active. Choosing **Cancel** during the READ cancels program execution. If SET DEVELOPMENT is OFF, the **Cancel** option on the **Program** menu is disabled during a READ.

The default value of SET DEVELOPMENT is ON.



# SHOW WINDOW

---

<b>Purpose</b>	Displays one or more user-defined windows or FoxPro system windows without activating them
<b>Syntax</b>	<b>SHOW WINDOW</b> <window name1> [, <window name2> ... ]   <b>ALL</b> [ <b>IN [WINDOW]</b> <window name3>   <b>IN SCREEN</b> ] [ <b>REFRESH</b> ] [ <b>TOP   BOTTOM   SAME</b> ] [ <b>SAVE</b> ]
<b>See Also</b>	ACTIVATE WINDOW, DEFINE WINDOW

**Description** SHOW WINDOW controls the display and front-to-back screen placement of windows. If windows are defined as hidden or haven't been activated, SHOW WINDOW displays the windows without activating them.

You can also display system windows, such as the Command window, the Calculator, the Calendar/Diary, and so on. Desk accessories (the Filer, the Calculator, and so on) must be active before they can be displayed. If one or more windows are currently displayed, SHOW WINDOW lets you change the front-to-back order of the windows.

You can't specify where output is directed with SHOW WINDOW. Use ACTIVATE WINDOW to direct output to a window.

**Clauses**      <window name1> [,<window name2> ... ]  
Specify the name of one or more windows to display with <window name1>, <window name2>, and so on.

**ALL**  
Include ALL to display all windows.

**IN [WINDOW] <window name3>**  
If IN WINDOW <window name3> is included, the window is displayed inside a parent window without assuming the characteristics of the parent window. A window activated inside a parent window can't be moved outside the parent window. If the parent window is moved, the child window moves with it.

The parent window specified with <window name3> must first be created with DEFINE WINDOW.

## **IN SCREEN**

By including **IN SCREEN**, you can explicitly place a window on the desktop in FoxPro for MS-DOS or the main FoxPro window in FoxPro for Windows instead of in another window. Windows are placed on the desktop or main FoxPro window by default.

## **REFRESH**

You can include the **REFRESH** clause to redraw a Browse window. This is useful on a network to ensure that you are browsing the most current version of a table/.DBF. The work area for the Browse window table/.DBF is selected.

Memo editing windows are refreshed with changes made to the memo field by other users on a network. **SET REFRESH** determines the interval between memo editing window refreshes. Refer to **SET REFRESH** in this manual for additional information on how data is refreshed in tables/.DBFs opened for shared use on a network.

## **TOP**

Include **TOP** to place the specified window in front of all other windows.

## **BOTTOM**

Include **BOTTOM** to place the specified window behind all other windows.

## **SAME**

**SAME** affects only windows that have been previously displayed or activated and then cleared from the desktop or main FoxPro window with **DEACTIVATE WINDOW**. Issuing **SHOW WINDOW SAME** places the specified window back into a stack of windows in the same position the window occupied before it was deactivated.

## **SAVE — FoxPro for MS-DOS only**

Including **SAVE** keeps an image of a window on the desktop or in another window after the window has been hidden. Normally, windows are removed from the desktop after they are hidden. The window image can be cleared from the desktop or a window with **CLEAR**. You can't save an image of the current output window.

**SAVE** is ignored in FoxPro for Windows.

**Example**

In this example, a window named Output is created and displayed. Since SHOW WINDOW is used to display the window, output can't be directed to the window until it is activated.

```
CLEAR  
DEFINE WINDOW output FROM 2,1 TO 13,75 TITLE 'Output' ;  
    CLOSE FLOAT GROW SHADOW ZOOM  
SHOW WINDOW output
```

# SYS(20)

Purpose	Transforms German text
Syntax	<b>SYS(20)</b>
Returns	Character
See Also	See the following description.

## SYS(20, <expC>, <expN>)

SYS(20) is used to order fields containing German words and names. The fields are ordered to resemble the ordering in a German phone book — upper- and lower-case are intermixed, eszets sort immediately after double s and characters with umlauts sort immediately after the unadorned character followed by the letter e. The following table lists the sort order of a sample set of characters:

A	C	T
a	c	t
AD	...	U
AE	r	u
ae	S	ua
Ä	s	ue
ä	sa	Ü
af	ss	ü
B	ß	V
b	sz	...

SYS(20) transforms the character expression <expC>. The numeric expression <expN> specifies the number of characters in <expC> to transform, starting with the first character in <expC>. Any additional characters after the position specified by <expN> are ignored. The length of character string returned by SYS(20) is 2 \* <expN> + 10 characters, with a maximum length of 254 characters.

In the following example, the CONTACT field contains the names of customers. This field is indexed using SYS(20) and the first 20 characters in the field. An structural index tag named GCONTACT is created.

```
USE CUSTOMER
INDEX ON SYS(20, contact, 20) TAG gcontact
```

# USE

---

**Purpose** Opens a table/.DBF file and associated index files

**Syntax** `USE [<file> | ?]  
[IN <expN1>]  
[AGAIN]  
[INDEX <index file list> | ?  
[ORDER [<expN2> | <idx index file>  
| [TAG] <tag name> [OF <cdx file>]  
[ASCENDING | DESCENDING]]]  
[ALIAS <alias>]  
[EXCLUSIVE]  
[SHARE]  
[NOUPDATE]`

**See Also** CREATE, DBF( ), USED( )

**Description** USE opens a table/.DBF file in the current work area.

If USE is issued without a table/.DBF name and a table/.DBF file is open in the current work area, the table/.DBF is closed. A table/.DBF is also closed when another table/.DBF is opened in the same work area.

**Clauses** `<file> | ?`

The name of the table/.DBF to open is specified with `<file>`. The Open dialog appears with a list of available tables/.DBFs to choose from if you include `?` instead of a table/.DBF name.

**IN <expN1>**

You can open a table/.DBF in a non-current work area by specifying the work area number with `<expN1>`. You can close a table/.DBF in a non-current work area by issuing USE without a file name and specifying the work area number.

The IN clause supports 0 as a work area. Including 0 opens a table/.DBF in the lowest available work area. For example, if tables/.DBFs are open in work areas one through ten, the command

```
USE customer IN 0
```

opens the client table/.DBF in work area 11.

This single command is equivalent to issuing these two commands:

```
SELECT 0  
USE customer
```

## **AGAIN**

To open a table/.DBF concurrently in multiple work areas, you can do one of the following:

- Select another work area and issue USE with the table/.DBF name and the AGAIN clause.
- Issue USE with the table/.DBF name and the AGAIN clause, and specify a different work area.

When you use a table/.DBF again in another work area, the table/.DBF in the new work area takes on the attributes of the table/.DBF in the original work area. For example, if a table/.DBF is opened for read-only or exclusive access and is used again in another work area, the table/.DBF is opened for read-only or exclusive access in the new work area.

Index files opened for the original table/.DBF are available for the table/.DBF you open again if you don't open indexes when you reopen the table/.DBF. The index order is set to 0 in the work areas where the table/.DBF is reopened.

You can open indexes that weren't opened with the original table/.DBF. This sets the index order to 0 for the original table/.DBF.

A reopened table/.DBF is assigned the default alias of the work area. You can include an alias every time you open a table/.DBF in multiple work areas as long as the aliases are unique.

## **INDEX <index file list> | ?**

You can open index files with the table/.DBF by including INDEX and specifying a set of indexes with <index file list>. If a table/.DBF has a structural compound index file, the index file is automatically opened with the table/.DBF. You can open a single index file by issuing INDEX ?. The Open dialog appears with a list of available index files.

<index file list> can contain any combination of names of .IDX and .CDX index files. You don't have to include the file name extensions for index files unless an .IDX and a .CDX index file in the index file list have the same name.

The first index file named in the index file list is the controlling index file, which controls how records in the table/.DBF are accessed and displayed. If the first index file is a .CDX compound index file, records in the table/.DBF are displayed and accessed in physical record order.

### **ORDER [<expN2>]**

Include the ORDER clause to specify a controlling index file or tag other than the first index file in <index file list>. <expN2> specifies the controlling tag or index file as it appears in the index file list. .IDX index files are numbered first in the order in which they appear in the index file list. Tags in the structural compound index file (if one exists) are then numbered in the order in which the tags were created. Finally, tags in any independent compound index files are numbered in the order in which they were created. You can also use SET ORDER to specify the controlling index file or tag. See SET ORDER in this chapter for a further discussion of the numbering of index files and tags.

If <expN2> is 0, records in the table/.DBF are displayed and accessed in physical record order, and the indexes remain open. Including ORDER 0 enables open index files to be updated while presenting the file in record number order. Including ORDER without <expN> is identical to including ORDER 0.

### **ORDER [idx index file]**

#### **ORDER [[TAG <tag name>] [OF <cdx file>]]**

You can also designate an .IDX index file as the controlling index file by specifying its name with <idx index file>.

To designate a tag of a .CDX index file as the controlling tag, specify the tag name with <tag name>. The tag name can be from the structural compound index file or any open compound index file. If identical tag names exist in open compound index files, include OF <cdx file> and specify the name of the compound index file containing the desired tag.

### **ASCENDING | DESCENDING**

Include the ASCENDING or DESCENDING key word after the ORDER clause to specify whether the table/.DBF records are accessed and displayed in ascending or descending order. Including one of these key words doesn't change the index file or tag; it alters only the order in which records are displayed and accessed.

## **ALIAS <alias>**

Include **ALIAS <alias>** to create an alias for the table/.DBF. You can refer to a table/.DBF by its alias in commands and functions that require or support an alias.

When a table/.DBF is opened, it is automatically assigned an alias, which is the table/.DBF name if **ALIAS** isn't included. You can create a different alias for the table/.DBF by including **ALIAS** and a new alias. An alias can contain up to 10 letters, digits or underscores and must begin with a letter or an underscore.

A default alias is assigned automatically if you open a single table/.DBF simultaneously in multiple work areas with **AGAIN** and you don't specify an alias when you open the table/.DBF in each work area.

A default alias is also assigned if a conflict occurs. For example:

<code>CLOSE DATABASES</code>	
<code>ACTIVATE WINDOW View</code>	<code>&amp;&amp; Open the View Window</code>
<code>USE customer ALIAS invoices IN 1</code>	<code>&amp;&amp; Alias is INVOICES</code>
<code>USE invoices IN 3</code>	<code>&amp;&amp; Conflict, alias is C</code>

## **EXCLUSIVE**

Including **EXCLUSIVE** opens a table/.DBF for exclusive use on a network. See **SET EXCLUSIVE** for more information on the exclusive use of tables/.DBFs.

## **SHARE**

Including **SHARE** opens a table/.DBF for shared use on a network. **SHARE** allows you to open a table/.DBF for shared use even when **EXCLUSIVE** is set **ON**. See **SET EXCLUSIVE** for more information on exclusive and shared use of tables/.DBFs.

## **NOUPDATE**

Including **NOUPDATE** prevents changes to the table/.DBF its structure.



**Example**

This example opens three tables in three different work areas. The View window is opened to show where the tables are open and their aliases.

```
CLOSE DATABASES
ACTIVATE WINDOW View
SELECT A
USE customer IN 0
USE invoices IN 0
USE salesman IN 0
```

## **New Generator Directives**

---

FoxPro includes these new generator directives:

- **#INSERT** — Inserts the contents of a file into code generated by GENSCRN or GENMENU.
- **#NAME** — Names a snippet produced by clauses such as SHOW, WHEN and VALID, whether the snippet is produced at the GET object or READ levels.
- **#NOREAD** — Prevents the generation of a READ statement.
- **#WCLAUSES** — Specifies additional DEFINE WINDOW clauses.

The following pages describe these directives in detail.

## #INSERT

---

<b>Purpose</b>	Inserts the contents of a file into code generated by GENSCRN or GENMENU
----------------	--

<b>Syntax</b>	<b>#INSERT &lt;file&gt;</b>
---------------	-----------------------------

---

**Description** The #INSERT generator directive inserts the contents of <file> into generated code.

Use #INSERT to include a file in several screen or menu programs. For example, use #INSERT to include a file of #DEFINE statements in several screen programs.

The directive can appear in any snippet; it does not have to be in a particular snippet, such as the setup code snippet. Additionally, the directive can appear anywhere in a snippet.

**Clauses** <file>

This must be a text file. The generator searches for the file in the current directory, in the FoxPro directory and subdirectories, and along the MS-DOS path. If the file cannot be found, the generator comments your code — indicating that the file could not be found — and then continues the generation process.

This file cannot include other screen generator directives.

# #NAME

---

Purpose	Assigns a name to a snippet produced by a clause such as SHOW, WHEN and VALID
Syntax	#NAME <snippet name>
Description	Names a snippet produced by clauses such as SHOW, WHEN and VALID, whether the snippet is produced at the GET object or READ level. The #NAME directive must appear at the top of a snippet, before any code statements. Blank lines or comments can precede the #NAME directive.
Clauses	<snippet name> This is the name assigned to the snippet. The <snippet name> must conform to the rules for naming procedures and functions.

FoxPro does not check for duplicate names.

# #NOREAD

---

<b>Purpose</b>	Prevents the generation of a READ command
<b>Syntax</b>	<b>#NOREAD</b>
<b>Description</b>	<p>Include the #NOREAD generator directive in a screen setup snippet to prevent the generation of the READ command.</p> <p>When #NOREAD is included in a screen setup snippet, only @ ... SAY GET, @ ... GET, and @ ... EDIT commands are generated. When you execute the screen code, the controls are displayed but are not activated.</p>

## #WCLAUSES

---

<b>Purpose</b>	Specifies additional DEFINE WINDOW clauses
<b>Syntax</b>	<b>#WCLAUSES</b> <clause1>,<clause2>, . . .<clausen>
<b>See Also</b>	_#READCLAUSES

**Description** The #WCLAUSES generator directive specifies additional DEFINE WINDOW clauses.

The #WCLAUSES directive must be in the setup snippet of the screen whose window definition you want to modify. If you do not check the **Define Windows** check box in the Generate Screen dialog, #WCLAUSES has no effect.

The setup snippet can contain only one #WCLAUSES directive. If more than one #WCLAUSES directive exists, FoxPro uses only the first one.

## New System Memory Variables

---

System memory variables are “built-in” memory variables that FoxPro creates and maintains automatically. System memory variables are by default PUBLIC; you can declare a system memory variable PRIVATE.

The following table lists the FoxPro system memory variables that are new for version 2.5 and their default values.

Variable	Type	Default Value
_DOS	L	.T. in FoxPro for MS-DOS
_MAC	L	.T. in FoxPro for Macintosh <sup>®</sup>
_TRANSPORT	C	TRANSPORT.PRG
_UNIX	L	.T. in FoxPro for UNIX <sup>®</sup>
_WINDOWS	L	.T. in FoxPro for Windows

Detailed descriptions of these variables appear in the following pages.

## **\_DOS**

---

<b>Purpose</b>	Contains true (.T.) if you are using FoxPro for MS-DOS
<b>Syntax</b>	<b>_DOS = &lt;expL&gt;</b>
<b>Remarks</b>	New to FoxPro 2.5
<b>See Also</b>	_MAC, STORE, _UNIX, VERSION( ), _WINDOWS

---

**Description** The \_DOS system memory variable contains a logical true (.T.) if you are using FoxPro for MS-DOS. \_DOS contains a logical false (.F.) if you are using a different FoxPro version (FoxPro for Macintosh, FoxPro for UNIX or FoxPro for Windows).

The value contained in \_DOS cannot be changed with STORE or =.



## **\_MAC**

---

<b>Purpose</b>	Contains true (.T.) if you are using FoxPro for Macintosh
<b>Syntax</b>	<b>_MAC = &lt;expl&gt;</b>
<b>Remarks</b>	New to FoxPro 2.5
<b>See Also</b>	_DOS, STORE, _UNIX, VERSION( ), _WINDOWS

---

**Description** The \_MAC system memory variable contains a logical true (.T.) if you are using FoxPro for Macintosh. \_MAC contains a logical false (.F.) if you are using a different FoxPro version (FoxPro for MS-DOS, FoxPro for UNIX or FoxPro for Windows).

The value contained in \_MAC cannot be changed with STORE or =.

## **TRANSPORT**

---

<b>Purpose</b>	Specifies the program to transport FoxPro 2.0 screens, labels and reports to FoxPro 2.5 formats
<b>Syntax</b>	<b><code>_TRANSPORT = &lt;program name&gt;</code></b>
<b>Remarks</b>	New to FoxPro 2.5
<b>See Also</b>	CREATE REPORT, CREATE SCREEN

**Description** With `_TRANSPORT` you can specify the program FoxPro 2.5 uses when it transports FoxPro 2.0 screens, labels and reports to FoxPro 2.5 formats. By default, `_TRANSPORT` contains `TRANSPORT.PRG`, the screen, label and menu conversion program installed in your FoxPro 2.5 directory.

The program you specify with `_TRANSPORT` runs when you attempt to open a FoxPro 2.0 screen, label or report in FoxPro 2.5. To specify a program other than `TRANSPORT.PRG` to transport your screens, labels and reports, store the program's name in `_TRANSPORT`. If your conversion program is in a directory other than the current default directory, include the path with the program name.

If you rename `TRANSPORT.PRG` or move it to another directory, store the new file name and/or its directory in `_TRANSPORT`.

You can also specify a conversion program in your FoxPro 2.5 configuration file by including the following line:

```
_TRANSPORT = <file name>
```

<file name> specifies the name of your conversion program and can include a path.

If you write your own transporter, it must accept one parameter — the name of the report, label or screen form to transport. Your transporter must return one parameter. The parameter can be one of the following values:

Return Code	Meaning
1	Successfully transported.
2	Opened without transporting.
3	Transportation cancelled.

For more information about transporting FoxPro 2.0 screens, labels and reports, refer to the Using Files From Other Platforms chapter.

## **\_UNIX**

---

<b>Purpose</b>	Contains true (.T.) if you are using FoxPro for UNIX
<b>Syntax</b>	<b>_UNIX = &lt;expL&gt;</b>
<b>Remarks</b>	New to FoxPro 2.5
<b>See Also</b>	_DOS, _MAC, STORE, VERSION( ), _WINDOWS

---

**Description** The \_UNIX system memory variable contains a logical true (.T.) if you are using FoxPro for \_UNIX. \_UNIX contains a logical false (.F.) if you are using a different FoxPro version (FoxPro for MS-DOS, FoxPro for Macintosh or FoxPro for Windows).

The value contained in \_UNIX cannot be changed with STORE or =.

## **\_WINDOWS**

---

<b>Purpose</b>	Contains true (.T.) if you are using FoxPro for Windows
<b>Syntax</b>	<b>_WINDOWS = &lt;expL&gt;</b>
<b>Remarks</b>	New to FoxPro 2.5
<b>See Also</b>	_DOS, _MAC, STORE, _UNIX, VERSION( )

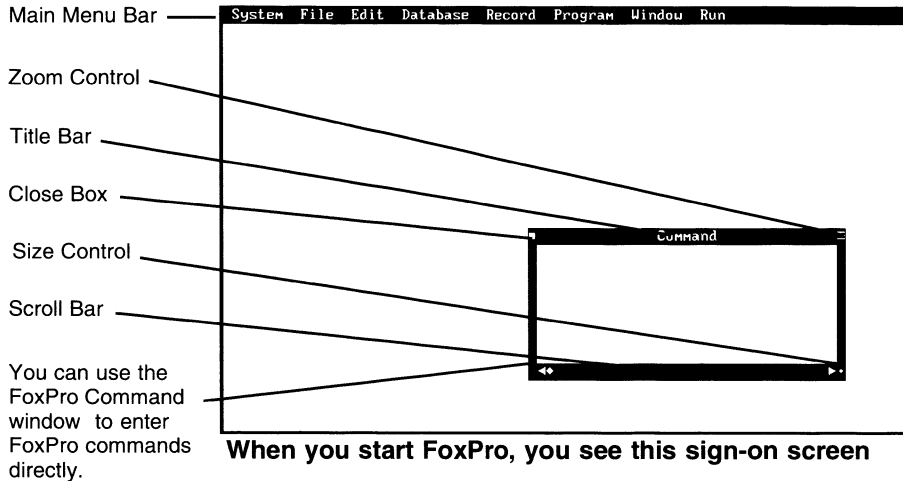
**Description** The \_WINDOWS system memory variable contains a logical true (.T.) if you are using FoxPro for Windows. \_WINDOWS contains a logical false (.F.) if you are using a different FoxPro version (FoxPro for MS-DOS, FoxPro for Macintosh or FoxPro for UNIX).

The value contained in \_WINDOWS cannot be changed with STORE or =.



## 2 Quick Start

FoxPro 2.5 provides the ease-of-use of a Graphic User Interface (GUI) on your character-based PC. Menus and dialogs make it easy to get the information you need when you want it.



This is a tip. You'll find a lot more of them sprinkled in the margins through the rest of this chapter. Usually they'll be clarifying something in the text or something shown in a graphic. Sometimes they'll be a hint about something that's coming up soon. You'll find it helpful to read them.

With FoxPro 2.5, you'll soon be running reports and applications, printing labels, browsing through your database files and getting quick answers to complex business questions. You'll even be able to create your own business applications. And you'll be doing all this even if you've never used a database management system before!


If you haven't done so already, please install FoxPro following the instructions in the FoxPro *Installation and Configuration* manual. Then, start FoxPro by moving into the FOXPRO25 directory and entering `FOX` at the DOS prompt.


In this chapter, we'll use a special menu, called the **Run** menu, to explore the many features of FoxPro. If you want to leave FoxPro at any point during this tour, use the mouse to choose **Quit** from the **File** menu (or press `Alt+F`, then `Q`).

# Some Navigation Aids

If you've worked with Windows or the Macintosh, you already know how to get around in FoxPro with a mouse. If you prefer not to take your fingers off the keyboard, that's okay, too, because we'll be telling you about our keyboard shortcuts.

Window	
Hide	
Clear	
Move	^F7
Size	^F8
Zoom ↑	^F10
Zoom ↓	^F9
Cycle	^F1
Color...	
Command	^F2
Debug	
Trace	
View	

 = "I changed my mind."

 = HELP

FoxPro windows work similarly to those in Microsoft Windows and other graphic environments. For example, to move the Command window:

- With the mouse, point to the window's title bar. Hold down the left mouse button as you drag the window to a new position, then release the button.
- With the keyboard, activate the menu bar by pressing the Alt key. Press the highlighted letter of the menu you want (W), then the letter of your menu option (M).

The Command window flashes to indicate that you can move it. Use the arrow keys. When the window is where you want it, press Enter.

If you haven't already, try moving the Command window.

Take a look at the Command window. You see a close box, zoom control, scroll bar and size control, just like under Windows or on the Macintosh. Since we won't be using the Command window in this booklet, close it. Click on the **Close** box or press Alt+F then C.

FoxPro also includes one-step keyboard shortcuts, shown next to most menu selections. For example, Ctrl+F2 makes the Command window reappear after you've closed it. To use the Ctrl key, hold it down, then press the letter or function key that does what you want.

Open the **Edit** menu now (click on **Edit** or press Alt+E). You'll see that we've used the same letters for shortcuts as some Macintosh and Windows programs. This makes FoxPro easier to learn and use.

Now, press the Escape key to leave the **Edit** menu without making a selection. The Escape key can be used whenever you change your mind about making a selection or taking an action.

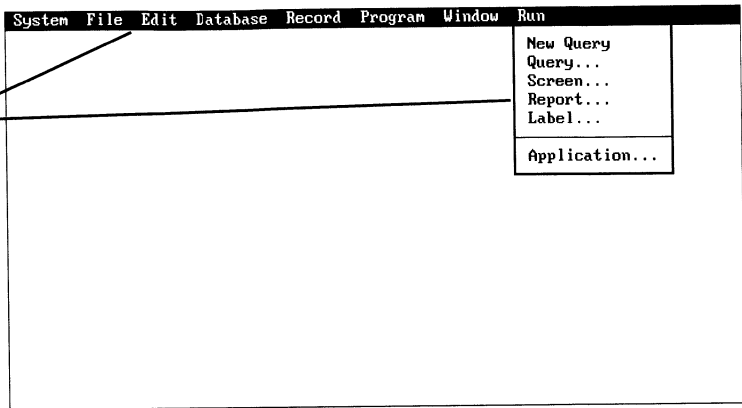
As we get further into using FoxPro, you may want more information than we give you in this book. If so, just press the F1 key and FoxPro will provide help on whatever you're doing at the time (that's our "context-sensitive" help in action).



## FoxPro Makes It Easy to Find Information

---

You don't have to be a programmer with FoxPro because all of the important tasks you frequently want to do are available from the menu system.



Open the **Run** menu by clicking once on **Run** or pressing Alt+N.

This menu provides access to the major features you want in any database management system. You want to see useful information quickly and easily. FoxPro lets you get it, how and when you want it.

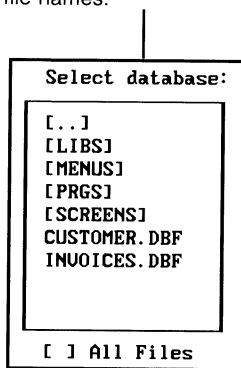
The **Run** menu provides access to:

- **Queries** — Database searches that give you answers to business question. Queries are created in the RQBE window. RQBE creates search requests in a form that FoxPro understands. You can save them to re-use whenever you want.
- **Screens** — Screen layouts used for entering and editing information in your database files. They can be made to look just like printed forms that you fill out by hand.
- **Reports** — Collections of data that are organized, summarized or otherwise manipulated to present the data in your files in a more informative manner. Reports can be viewed on the screen, sent to a printer, or saved as a disk file.

Listed directories and subdirectories (similar to Macintosh folders) are shown in square brackets: **[directory name]**.

The **[..]** is MS-DOS for the parent directory of the directory you are in.

Unbracketed items are file names.



- **Mailing Labels** — A very common use for a DBMS like FoxPro. In FoxPro, lay out your labels just the way you want them, then save the format by name so you can use it over and over again.
- **Applications** — Programs used to do the same jobs over and over simply and easily. An application might perform payroll functions, engineering analysis, and so on.

RQBE is short for Relational Query By Example, a mouthful which sounds more mysterious than it really is. RQBE is just a computer scientist's term for a way of quickly getting the business answers you need, whether or not you're a programmer. Spend a bit of time on it now and you'll save hours and days, or more, later. It's a very powerful information tool that is very easy to use.

Actually, RQBE is no more difficult to learn and use than a new clock radio — and certainly easier than a new VCR. With a new clock radio, you'd spend a few minutes looking at the buttons and knobs and reading the labels, then you'd just use it. You can use RQBE just as easily.

Later we'll show you just about everything you can do with RQBE, but for now let's see how easy it is to browse through a database file for the information you want.

Choose **New Query** from the **Run** menu. If no database files are open, you'll see a dialog that allows you to choose the one you want. You can do several things in this dialog, but we only need to work with the list at the left.

The file we want is in the TUTORIAL subdirectory, so select it now. Tab to the list, if necessary, highlight **[TUTORIAL]** using the arrow keys, then press Enter. Or, just double-click on **[TUTORIAL]**. From the new list you see, select CUSTOMER.DBF the same way.

This takes you to the RQBE window shown on the next page.

## Three Steps to RQBE

1. Choose the database files that RQBE uses.

2. Choose the fields that RQBE displays.

3. “Press” the **Do Query** push button. FoxPro searches the database(s) and displays your RQBE answer.

The screenshot shows the RQBE - UNTITLED window. It has three main sections: Databases, Output Fields, and Output To. The Databases section has a list box with 'CUSTOMER' selected and buttons '< Add >' and '< Clear >'. The Output Fields section has a list box with 'CNO', 'COMPANY', 'CONTACT', 'ADDRESS', 'CITY', and 'STATE' selected and buttons '[X] Select Fields...', '[ ] Order By...', '[ ] Group By...', and '[ ] Having...'. The Output To section has a 'Browse' button, '[ ] Options...', '< See SQL >', and '<< Do Query >>'. Below these sections is a table with columns 'Field Name', 'NOT', 'Example', 'Up=Lo', and 'Select Criteria'. The table is currently empty.

The RQBE window contains a lot because it does a lot. For now just ignore most of what you see. All we need is three of the “clock-radio-like” functions of RQBE.

If the Databases list is empty, press **Add** then select CUSTOMER.DBF.

If other files are listed, remove them. Highlight each one, then press **Clear**.

With the keyboard, tab to the list box, then use the arrow keys to highlight the file name you want. Press the Spacebar to select it, then tab down and press **Clear**. Use Shift+Tab to get back to the list if you want to remove more than one file.

At the left of the window, you see the names of one or more database files. In this case, only the CUSTOMER database appears.

Next is a list of field names, or column headings, for the data table.

And last, we need the **<<Do Query>>** push button. It's the default button, enclosed in double brackets (rather than single brackets like **<Clear>**). Push it now by pressing Ctrl+Enter or just clicking on it.

Your screen should look like the picture at the top of the next page. You're looking at a simple table of information, yet this is the basis of every relational database management system (DBMS) in existence. (We told you it wasn't mysterious.)

You've probably worked with similar tables in a spreadsheet or word processor. You set up columns and label them at the top, then enter your information row by row below that. In a DBMS, the headings are called “fields” and the rows of information are called “records.”

## Browsing a Database File

Columns are called "fields" while rows are called "records."

FoxPro presents your query answer in a

Change column sizes by dragging the vertical lines or using the **Browse** menu.

System File Edit Database Record Program Window Run Browse			
QUERY			
Cno	Company	Contact	Output To
14021	1st Computers	Jeff W. Culbert	<input type="button" value="Browse"/>
18232	1st Data Reductions	Dennis Johnson	
12082	1st Software Systems Ltd.	Rance Sivren	[ ] Options...
12840	1st Survey	Robert Hepworth	< See SQL >
A8872	A Beck Pertamina	Jim Ansarti	<< Do Query >>
A8818	A. Arts Computers	Darryl Roudebus	Up=Lo Select
A6459	A. Bloomington Biz	Phil Putnam	Criteria
A6188	AZ Inc	Tom Totah	<Insert>
A5181	Abbymark Uelonex	Isador Sweet	<Remove>
A3964	Acres Tree Solutions	Russell Kmickle	< Or >
A3882	Add Associates	Len Silverman	
A1046	Add Inc	Bert Crauford	
A7249	Adder Incorporated	Brenda Carturig	
A3835	Adv. Software	Barbara H. Mart	

You can zoom the Query window. Click on the upper right corner or choose **Zoom** from the **Window** menu. You can scroll vertically or horizontally to see more of your information. Click on the scroll bars or use Tab and Shift+Tab, PgDn and PgUp.

You can change the order of the fields by dragging the column headings (field names) where you want them. If you don't have a mouse, use the **Move Field** option on the **Browse** menu.

You can even change the size of the columns by moving the lines between the fields. Drag the lines or use the **Size Field** option on the **Browse** menu.

Try manipulating the Query window. It doesn't let you change the data, so you can't hurt your database.

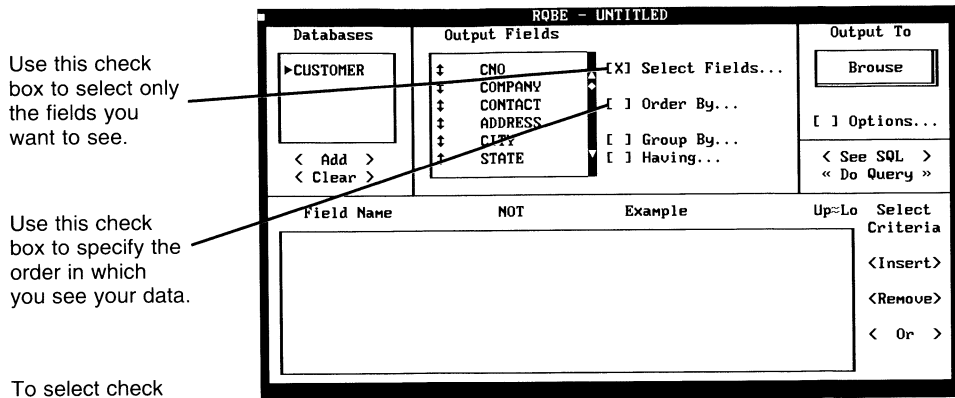
Until now, we've been browsing through all the data in the CUSTOMER database. With RQBE, you can also select only the data you want to see. We'll do that next.

Press Escape to close the Query window and return to the RQBE window. We're going to simplify this information next.

When it finished working, FoxPro told you how many records it found and how long it took to find them. That message goes away when you start moving around on the screen.

Browse	
Change Grid Off	
Unlink Partitions	
Change Partition ^H	
Size Field	
Move Field	
Resize Partitions	
Goto...	
Seek...	
Toggle Delete	^T
Append Record	^N

## Turning Data into Information



To select check boxes, push buttons and radio buttons: Tab until the control is highlighted, then press Enter. Or just click on the control with the mouse.

FoxPro check boxes appear as square brackets. You can select as many as you want from a group.

[X]	Wrap word
[X]	Auto indent
[X]	Add line feeds

Database tables typically contain more information than you want to see all the time. If we wanted to look up customer phone numbers we wouldn't need the customer ID number (CNO), the address, and so on. We can fix this by simply picking only the fields we want and we'll do that now.

Choose the **Select Fields...** check box, then choose **Remove All**. This empties the **Selected Output** list so you can select the specific fields you want.

Select the CUSTOMER.COMPANY, CUSTOMER.CONTACT, CUSTOMER.PHONE and CUSTOMER.STATE fields. To select them one at a time, highlight a field and press the Spacebar or just click on the field. When the field is marked with a triangle, click **Move→** or press the "M" key on the keyboard.

To move several fields at once, hold down the Shift key as you make your choices, then **Move→** them. Or just double-click on each field you want. Choose **OK** when you're done to get back to the RQBE window.

Database Fields				Selected Output	
CUSTOMER.COMPANY	C	< Move + >	↑	CUSTOMER.COMPANY	
CUSTOMER.CONTACT	C	< All + >	↑	CUSTOMER.CONTACT	
CUSTOMER.ADDRESS	C	< Remove >	↑	CUSTOMER.PHONE	
CUSTOMER.CITY	C	< Remove All >	↑	CUSTOMER.STATE	
CUSTOMER.STATE	C				
CUSTOMER.ZIP	C				
CUSTOMER.PHONE	C				
CUSTOMER.ONO	C				

Functions/Expressions			
Functions		<input type="checkbox"/> No Duplicates <input type="checkbox"/> Cross Tabulate	
		« OK » < Cancel >	

Choose **Do Query** to see your phone list. If the Query window is zoomed, you can see all the fields. If the fields aren't in the order you want, just move them around.

QUERY			
Company	Contact	Phone	State
1st Computers	Jeff W. Culbertson	617/232-5853	MA
1st Data Reductions	Dennis Johnson	584/524-3966	LA
1st Software Systems Ltd.	Rance Sivren	713/723-1288	TX
1st Survey	Robert Hepworth	214/243-7247	TX
A Beck Pertamina	Jim Ansarti	617/643-6928	MA
A. Arts Computers	Darryl Roudebush	617/662-8157	MA
A. Bloomington Biz	Phil Putnam	984/222-9457	FL
AZ Inc	Tom Totah	617/823-5188	MA
Abbymark Uelonex	Isador Sweet	214/922-4927	TX
Acres Tree Solutions	Russell Kwickle	281/786-8785	NJ
Add Associates	Len Silverman	415/897-2818	CA
Add Inc	Bert Crauford	383/499-2886	CO
Adder Incorporated	Brenda Cartwright	313/573-5873	MI
Adv. Software	Barbara H. Martin	617/646-7974	MA
Advantage Computer School	Duane Marshall	488/946-1317	CA
Aerial Inc.	Lynn Williams	281/696-7378	NJ
Alex County Community Corp	Rance Hayden	381/459-4484	MD
Alex Systems	Nancy Wright	814/838-3116	PA
American Computer Company	Dick W Guyton	886/799-7786	TX
American Forum	Gui Dupuy	885/682-5588	CA

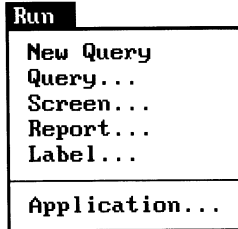
Now press Escape twice to close the Query and RQBE windows.

When asked if you want to save the changes, answer **Yes**. Then enter MYQUERY as its name to save it under in the next dialog and choose **Save**.

The next time you want to look at a phone list for the companies in your CUSTOMER database, you'll be able to do it by simply selecting your query by name using the **Run** menu.

We'll show you just how easy that is next.

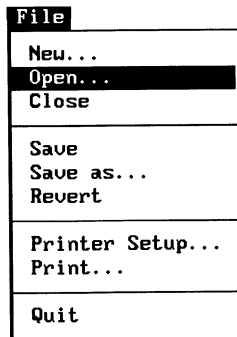
## Saved Queries Save You Time and Effort



With saved Queries, you ask the same questions but get answers based on the newest data in your files.

Saved Queries can be run again and again with ease. To look at the phone numbers we just saw, select **Query...** from the **Run** menu then choose MYQUERY.QPR from the file list. The information you want is yours, instantly and easily.

System File Edit Database Record Program Window Run Browse			
QUERY			
Company	Contact	Phone	State
1st Computers	Jeff W. Culbertson	617/232-5853	MA
1st Data Reductions	Dennis Johnson	584/524-3966	LA
1st Software Systems Ltd.	Rance Sivren	713/723-1288	TX
1st Survey	Robert Hepworth	214/243-7247	TX
A Beck Pertamina	Jim Ansarti	617/643-6920	MA
A. Arts Computers	Darryl Roudebush	617/662-0157	MA
A. Bloomington Biz	Phil Putnam	984/222-9457	FL
AZ Inc	Tom Totah	617/823-5180	MA
Abbymark Velonex	Isador Sweet	214/922-4927	TX
Acres Tree Solutions	Russell Kwickle	201/786-0785	NJ
Add Associates	Len Silverman	415/897-2810	CA
Add Inc	Bert Crauford	303/499-2086	CO
Adder Incorporated	Brenda Cartwright	313/573-5873	MI
Adv. Software	Barbara H. Martin	617/646-7974	MA
Advantage Computer School	Duane Marshall	408/946-1317	CA
Aerial Inc.	Lynn Williams	201/696-7378	NJ
Alex County Community Corp	Rance Hayden	301/459-4484	MD
Alex Systems	Nancy Wright	814/838-3116	PA
American Computer Company	Dick W Guyton	806/799-7706	TX
American Forum	Gui Dupuy	805/682-5580	CA



After you've worked with a Query for a while, you may find that you want to change it, just a bit. You can do that easily without having to start all over again.

Let's modify the example query to see your company contacts listed alphabetically by state. First close the Query window, then open the **File** menu. This menu is similar to what you'd see in Windows or on a Macintosh. Select **Open...**

FoxPro works with different types of information. To make it easier to find files, lists like the one in the Open File dialog are often limited to one kind of file.

To change the file type, use the popup (see next page) at the lower left of the dialog. When you choose **Query** from this popup you'll see only Query files (file names ending with a .QPR extension). Select MYQUERY.QPR, then choose **Open**.

## Changing Queries Is Easy

You are back to the familiar RQBE window with the setup you created earlier for MYQUERY. We want to change the order in which the records appear, so choose the **Order By...** check box. In the dialog, select CUSTOMER.STATE, then **Move** →. Choose **OK**.

CUSTOMER.STATE now has an upward pointing arrow next to it indicating that the query will be shown by state in ascending order, and a number 1 indicating that this is the primary ordering field. Choose **Do Query** to see the results.

System File Edit Database Record Program Window Run Browse			
QUERY			
Company	Contact	Phone	State
Vergen Endeavors	Tom Droege	907/543-3784	AK
Steven Computers	Scott Sanderson	205/881-2245	AL
Walker Business Gallery	Bob Vogeltanz	205/749-9081	AL
Harpoon Store	Ron Morrison	501/663-9706	AR
Goods For The Masses	Steven Hagerty	602/276-5827	AZ
Great Form	Dorit Mowery	602/892-8572	AZ
Harpoon PCA	John Bryant	602/943-5082	AZ
Pro and Power	Joel Miller	602/276-6239	AZ
Quik Assistance	Laurence Krouse	602/953-8069	AZ
Raybank Services & Computing	Jerry Campanonoy	602/943-4609	AZ
SC Co.	Gene Patterson	602/991-1550	AZ
Thralls Senski	John Mattheues	602/894-9367	AZ
Weichert Compuserve	Gary Cumming	604/381-2591	BC
Add Associates	Len Silverman	415/897-2810	CA
Advantage Computer School	Duane Marshall	408/946-1317	CA
American Forum	Gui Dupuy	805/682-5580	CA
Atec Data Service	Randy Keji	408/246-5353	CA
Automated Mayo Miley	Bill Hopkins	714/540-6062	CA
Azimuth Bavis & Systems	Chuck Heitmeier	415/989-1603	CA

FoxPro popup controls have double lines on the right and bottom.

Database

To display the contents, Tab to the popup and press Enter. Use the arrow keys to highlight your choice then press Enter.

With a mouse just click on the popup and drag to your choice, then release the mouse button.

When you're done, close the Query window.

If you'd like more practice with what you've learned so far, you can go to the Retrieving Your Data chapter in the FoxPro *Getting Started* manual or the Relational Query By Example (RQBE) chapter of the FoxPro *User's Guide*. RQBE is easy to use, and that's part of what makes it a FoxPro power tool.

You can come back to this when you're ready. For a change of pace, we're going to run an application next.

Close the RQBE window now and save the query with its changes.



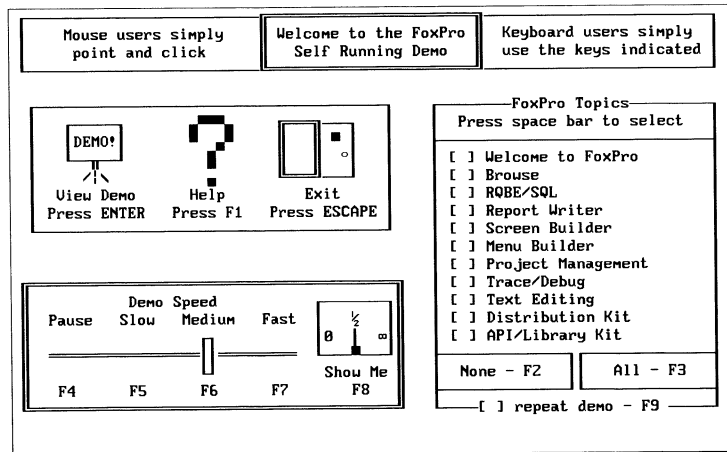
## Applications Can Make Tough Jobs Simple

An application is a large program that does a complicated job. You, your MIS department or an outside developer model your business operations to create applications that manage your information quickly, reliably and with the least amount of effort.

As an information user, you don't have to think about any of that. Just run the applications with FoxPro.

We'll run the self-running demonstration application that comes with FoxPro.

From the **Run** menu, choose **Application...**, then move to the FOXPRO25 subdirectory (choose **[.]**). Select DEMO.APP, then choose **Run**.



When you see the first screen of the demo, press F2 to clear the topic selections. Press F6 to select a medium speed. If you find this is the wrong speed later, you can press Escape to come back to the main screen and reset it. Or just press F1 then change the speed while the demo is running.

Check the **RQBE/SQL** check box.

Once the demo starts, you'll see a lot of familiar screens, as well as a few things you haven't seen yet. Don't bother memorizing any of this. At one point, you'll see the SQL (Structured Query Language) programming statement that RQBE creates to give you its answers so fast. If you want, you can enter statements like this yourself. (The *FoxPro Language Reference* tells you how.)

When you're ready, press Enter, sit back and watch the screen. You can return to the beginning of the demo at any time by pressing Escape.

The demo is a short overview to show you the kinds of things that can be done with RQBE. For more details, you can check the *FoxPro User's Guide* and the *FoxPro Language Reference*.

Now, check the **Browse** check box and uncheck the **RQBE/SQL** check box. Remember, you've already browsed through data using RQBE, knowing that RQBE lets you look at your data without any fear of accidentally changing the information. But you can also edit the information in your database files in Browse windows.

Press Enter to run the Browse demo now.

If you're interested in learning more about Browse, you'll find it on the **Database** menu. The Looking at Your Data chapter of *FoxPro Getting Started* teaches you more about it. A complete reference can be found in the Database Menu chapter of the *FoxPro User's Guide*.

This is a simple application, but it should give you a good idea of how easy it is to run *any* FoxPro application. A few clicks or keystrokes, then a name selection and it's yours.

One of the things you saw in the RQBE demo was some activity in the bottom half of the RQBE window. Next we'll show you how to use this area to select only the records you want to see from a database, and then create a report.

Press Escape to end the DEMO.APP application.

Database
Setup...
Browse
Append From...
Copy To...
Sort...
Total...
Average...
Count...
Sum...
Calculate...
Report...
Label...
Pack
Reindex

## It's Easy to Get Just the Data You Want

It's nice to be able to get to your data quickly and easily, but you won't want to see all of your data all of the time.

RQBE makes data selection easy. We'll use it to select only the customers in California, then show you how to instantly turn the query into a report that you can view or print whenever you want to.

The page header is repeated at the top of every page when you print your report.

These are called the detail lines and contain information from your database file.

The page footer is repeated at the bottom of every page when you print your report.

Company	Contact	Phone	State
1st Computers	Jeff W. Culbertson	617/232-5053	MA
1st Data Reductions	Dennis Johnson	504/524-3966	LA
1st Software Systems Ltd.	Rance Sivren	713/723-1288	TX
1st Survey	Robert Hepworth	214/243-7247	TX
A Beck Pertamina	Jim Ansarti	617/643-6920	MA
A. Arts Computers	Darryl Roudebush	617/662-0157	MA
A. Bloomington Biz	Phil Putnam	904/222-9457	FL
AZ Inc	Tom Totah	617/823-5180	MA
Abbymark Velonex	Isador Sweet	214/922-4927	TX
Acres Tree Solutions	Russell Kmickle	201/786-0785	NJ
Add Associates	Len Silverman	415/897-2810	CA
Add Inc	Bert Crawford	303/499-2086	CO
Adder Incorporated	Brenda Cartwright	313/573-5873	MI
Adv. Software	Barbara H. Martin	617/646-7974	MA
Advantage Computer School	Duane Marshall	408/946-1317	CA
Aerial Inc.	Lynn Williams	201/696-7378	NJ
Alex County Community Corp	Rance Hayden	301/459-4484	MD
Alex Systems	Nancy Wright	814/838-3116	PA
American Computer Company	Dick W Guyton	806/799-7706	TX
American Forum	Gui Dupuy	805/682-5580	CA
American Innovations	Garret Hill	201/245-7517	NJ
Ansari Data Software	Wallace Campanelli	412/931-0818	PA
Ansari Produce	Oliver Mossangan	914/357-6767	NY
Applied Telephone Realtors	Michael Cumming	215/359-2769	PA
Aspen Planning & Inc.	Gary Erickson	918/254-3104	OK
Aspen Technology	Mel Colby	319/659-8285	IA
02/03/92			Page 1

To save some effort, we'll start with MYQUERY and modify it. From the **File** menu, choose **Open....** In the dialog, use the **Type** popup to change the file type to **Query**. Change to the TUTORIAL subdirectory, then select MYQUERY.QPR and choose **Open**.

The RQBE window shows the fields selected and the order in which we'll see the records. Choose **Do Query** to take another quick look at the data. Zoom the Query window to fill the screen, then scroll down a few screens, scanning the data as you go.

This database contains information on companies in every state of the U.S. If we were planning a trip to the West Coast, we might want to take along a phone list of only the contacts in California. RQBE makes this easy.

Close the Query window to get back to the RQBE window.

You select data by telling FoxPro what information you want from which fields. First we need to specify the field we want to use. Click on the rectangle below the words Field Name. Release the mouse button when CUSTOMER.STATE is highlighted. (Or tab to the rectangle, then use the Spacebar and arrow keys to select the field.)

The **NOT** check box would be checked if we wanted all the customers *except* those in California, but we don't.

With RQBE, the only thing you have to do is select your fields and type in the values you want (or don't want) – database management with no programming at all.

Field Name	NOT	Example	Up=Lo	Select Criteria
‡ CUSTOMER.STATE	<input type="checkbox"/>	Like Exactly Like More Than Less Than Between In	<input type="checkbox"/>	<Insert> <Remove> < Or >

You can ask for searches like: customers whose year-to-date purchases are *More Than \$10,000* (entered as 10000 with no dollar sign, no comma), or customers *In CA, OH, NY* (use commas to separate the items).

Tab to the word **Like**, then press the Spacebar. (Or drag the word with the mouse.) This popup contains the conditions available for selecting the information RQBE looks for. For now, just leave **Like** selected and move to the area under Example.

We need an example (this *is* Relational Query By *Example*, after all) of what to include so type **CA** in the text box just below the word Example. The selection criteria tells FoxPro to “Find all of the records in which the CUSTOMER.STATE field looks like CA.” Notice that CA is entered in upper-case letters because all the records in the CUSTOMER table have upper-case states. If you want to match records regardless of case, choose the **Up=Lo** check box.

Choose **Do Query** and FoxPro gives you your answer in just a few seconds.

Next we'll show you how easy it is to take what FoxPro has done and turn it into a much more attractive report. Close the Query window now.

If you want to take a break, close the RQBE window. You can get to the RQBE window later select **Open...** from the **File** menu, and choosing MYQUERY.QPR.

## “Quick Reports” Really Are Quick and Easy

To print a report, uncheck the **Preview Report/Label** box, then check the **To Printer** box. Choose **OK**. Do the query to print the report.

If you use the **To File** check box, RQBE sends the report – headers, footers and all – to a text file you name. You can then modify the text with a word processor or page layout program.

To produce a report, first change the selection on the **Output To** popup (in the upper right corner of the RQBE window) from **Browse** to **Report/Label**. Check the **Options...** check box.

RQBE Display Options:

Formatting Options  
☐ Screen Display  
☒ Report  
☐ Label  
☐ Report/Label Form Name

☒ Quick Report... ☐ Overwrite

☒ Preview Report/Label  
☐ Show Summary Info Only  
☐ Eject Page Before Report  
☐ Report Heading

☐ Suppress Column Headings  
☒ Console On ☐ Console Off  
☒ Pause Between Screens

Output Destinations  
☒ To Printer  
☐ To File   
☐ Overwrite File  
☐ Use Printer Setup In Report/Label Form

< OK >  
 < Cancel >

In the RQBE Display Options dialog, select the **Report** radio button, then check the **Quick Report...** check box.

FoxPro radio buttons appear as parentheses. You can choose only one from a group.

( ) Left justify  
 (•) Right justify

RQBE Display Options:

Formatting Options  
☐ Screen Display  
☒ Report  
☐ Label

☒ Quick Report... ☐ Overwrite

☒ Preview Report/Label  
☐ Show Summary Info Only  
☐ Eject Page Before Report  
☐ Report Heading

☐ Suppress Column Headings  
☒ Console On ☐ Console Off  
☒ Pause Between Screens

RQBE Quick Report:  
☒ Column Layout  
☐ Form Layout

Report Width 80

Save as

Field1 Field2  
 xxxxxx xxxxxx  
 xxxxxx xxxxxx  
 xxxxxx xxxxxx

< OK > < Clear >

Output Destinations  
☐ To Printer  
☐ To File   
☐ Overwrite File  
☐ Use Printer Setup In Report/Label Form

< OK >  
 < Cancel >

Quick reports can be created using field names as column headings, or displaying field names and fields along the left side, with the information in one long column. Leave the **Column Layout** radio button selected and type TUTORIAL\MYQUERY.FRQ in the **Save as** text box. Choose **OK** and then **OK** again to return to the RQBE window

Now choose **OK** to exit the RQBE Quick Report dialog and **OK** again to exit the RQBE Display Options dialog. Choose **Do Query** in the RQBE window to preview your report!

FoxPro automatically created a page header using the field names from your database file.

The detail lines are the database records for companies in California.

To see more of the report, choose **More** or use the PgDn key.

System File Edit Database Record Program Window Run RQBE			
Company	Contact	Phone	State
Add Associates	Len Silverman	415/897-2818	CA
Advantage Computer School	Duane Marshall	408/946-1317	CA
American Forum	Gui Dupuy	805/682-5580	CA
Atec Data Service	Randy Keji	408/246-5353	CA
Automated Mayo Miley	Bill Hopkins	714/540-6062	CA
Azimuth Bavis & Systems	Chuck Heitmeier	415/989-1603	CA
Azimuth Corp	Al Reetz	619/271-8518	CA
Battery Weaver	Brian Case	415/952-7761	CA
Belmar Fishing Systems	Mike Ozer	415/323-2469	CA
Belmar Tronixs Computer Compuserve	Andy Rigney	213/452-9369	CA
Big Incorporated	Bert Dalglish	818/762-5886	CA
Blake Inc.	Bob Dot	415/992-0710	CA
Bob Lewis Walker & Jackson	Dan Tillotson	213/533-7332	CA
Bulldog Inc.	Bill Kane	415/864-7557	CA
Business Equipment Termi	Larry Van Lockern	619/564-2809	CA
C & S Computing & Systems	Parky Kwickle	415/325-4305	CA
« Done » < More > Column: 0			

Choose **More** (or press the PgDn key) a few times until you get to the next dotted line in the report. This indicates a page break.

Notice the page footer (with the date and, if you scroll to the right, the page number) that appears at the bottom of every page and the repeated page header — all done for you automatically by FoxPro.

But we can do even better than this without a lot of work. Choose **Done** to get back to the RQBE window.

## Letting the Report Writer Work for You

Now that we've got a report, we can make it more attractive using the Report Writer. It's easy. We'll change our report to look like this:

We'll change the repeating page header by repositioning the column headings and adding a box around them.

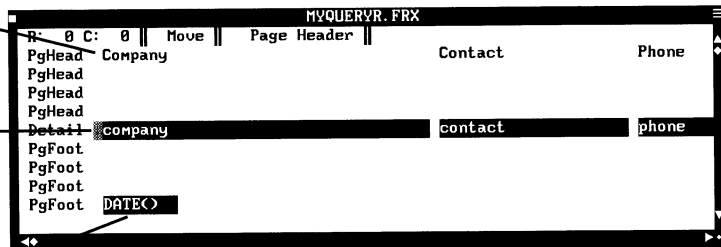
Company	Contact	Phone	State
Add Associates	Len Silverman	415/897-2810	CA
Advantage Computer School	Duane Marshall	408/946-1317	CA
American Forum	Gui Dupuy	805/682-5580	CA
Atec Data Service	Randy Keji	408/246-5353	CA
Automated Mayo Miley	Bill Hopkins	714/540-6062	CA
Azimuth Bavis & Systems	Chuck Heitmeier	415/989-1603	CA
Azimuth Corp	Al Reetz	619/271-8518	CA
Battery Weaver	Brian Case	415/952-7761	CA
Belmar Fishing Systems	Mike Ozer	415/323-2469	CA
Belmar Tronixs Computer	Andy Rigney	213/452-9369	CA
Big Incorporated	Bert Dalglish	818/762-5886	CA
Blake Inc.	Bob Dot	415/992-0710	CA
Bob Lewis Walker & Jackson	Dan Tillotson	213/533-7332	CA

To start, from the **File** menu select **Open....** Change the file type to **Report**, then select MYQUERY.FRX.

Text in the Page Header band.

Fields in the Detail band.

FoxPro function in the Page Footer band.



This screen shows the format of the quick report FoxPro created for you. Scroll to the right to see the rest of the header and footer. We'll change the header.

Text, fields and FoxPro functions are all “objects” that you can move around, just as you would in Microsoft Windows or on a Macintosh.

To move objects:

- With the mouse, select and drag the object.
- With the keyboard, use the arrow keys, Tab, and Shift+Tab to position the cursor on the object, press the Spacebar to select it, then use the arrow keys to move it. When the object is where you want it, press the Spacebar again.

*PgHead* stands for “page header.” Information placed here appears at the top of every page.

*Detail* stands for the information from your database file: the details from records.

*PgFoot* stands for “page footer.” Information placed here appears at the bottom of every page.

To start over, select the box, then press the Delete key.

Move “Company” in the page header so it’s centered above the company field, leaving a blank line between the title and the field. Then center the other column headings above the fields on the same line, too.

Boxes are drawn starting from the upper left corner of the box. To add a box around the column headings, first position the cursor or mouse pointer) to the left and one line above the “Company” heading.

Now choose **Box** from the **Report** menu. A small flashing box appears. Use the arrow keys or drag until the box is positioned about as shown below. When it’s where you want it, press the Spacebar or release the mouse button.

To see what your report is going to look like, choose **Page Preview...** from the **Report** menu. View the report, then close the Preview window. Save the report and close the Report Writer window, then close the RQBE window, too.

Preview			
Company	Contact	Phone	State
Add Associates	Len Silverman	415/897-2810	CA
Advantage Computer School	Duane Marshall	408/946-1317	CA
American Forum	Gul Dupuy	805/682-5580	CA
Atec Data Service	Randy Keji	408/246-5353	CA
Automated Mayo Miley	Bill Hopkins	714/540-6062	CA
Azimuth Bavis & Systems	Chuck Heitmeier	415/989-1603	CA

Next we’ll show you how to use multiple files in reports. You’ll find this feature useful, no matter what kind of business you’re in.



## Even Multi-File Reports Are Easy

So far, we've been working with a single database file, but you'll find that you often want to combine information from different databases into a single report. This used to be a big deal. You would have to "open files in multiple work areas," "set relations" and do various other complex things.

Not anymore. With RQBE you just join the files. As an example, you might keep customer names and addresses in one file, and customer invoices in another. To produce a report that shows the invoices by company, you'd need to combine information from both files.

The column headings that begin with an "I" are from the Invoices file, while the others are from the Customer file.

Repeating data in columns (like companies and contacts) can be suppressed.

Information can be grouped and totaled based on requirements that you specify.

03/28/92				Page 1
Company	Phone	Contact	Idate	Itotal
Atec Data Service	408/246-5353	Randy Keji	05/17/90	2721.19
			05/30/90	163.72
			05/26/90	441.91
			05/31/90	762.56
			05/28/90	744.49
				<hr/> 4833.87
Automated Mayo Miley	714/540-6062	Bill Hopkins	05/08/90	2336.34
				<hr/> 2336.34
Azimuth Corp	619/271-8518	Al Reetz	05/08/90	2047.08
				<hr/> 2047.08
Belmar Tronixs Computer	213/452-9369	Andy Rigney	05/10/90	2353.43
				<hr/> 2353.43

The files need something in common so that you can join them. Our CUSTOMER.DBF and INVOICES.DBF files both have a field called CNO. It contains an identification number so we can tell which invoice is whose without having to enter all the customer information over and over again. We'll use these files for our report.

Use the **File** menu to open the query MYQUERY.QPR. Choose the **Select Fields...** check box, then remove CUSTOMER.STATE from the **Selected Output** list. (Select the field then choose **Remove**.) Move CUSTOMER.PHONE (drag the double-headed arrow on the left) so it's below CUSTOMER.COMPANY, then choose **OK**.

In the RQBE window, choose **Add....** Select INVOICES.DBF then choose **Open**. To join the two files, we need to select the field that holds the customer number in both databases. On the left popup FoxPro has already selected INVOICES.CNO and on the right popup CUSTOMER.CNO. If you wanted to join the databases on different fields, you could select them from the popups, but this is the match we want. We want records where the customers match, so make certain the **NOT** check box is *not* checked. Make sure the condition is **Like**, then choose **OK**.

Now we can add the fields we want from the INVOICES file. Choose the **Select Fields...** box, then scroll down the **Database Fields** list and move INVOICES.IDATE then INVOICES.ITOTAL to the Selected Output box. Choose **OK**. The RQBE window should look like this:

The screenshot shows the RQBE - MYQUERY.R window. It has three main sections: Databases, Output Fields, and Output To. The Databases section shows 'CUSTOMER' and 'INVOICES' selected. The Output Fields section shows a list of fields: COMPANY, PHONE, CONTACT, IDATE, and ITOTAL. The Output To section shows 'Report/Label' selected. Below these sections is a table for join criteria.

Field Name	NOT	Example	Up=Lo	Select Criteria
INVOICES.CNO	<input type="checkbox"/> Like	CUSTOMER.CNO	<input type="checkbox"/>	<Insert>
CUSTOMER.STATE	<input type="checkbox"/> Like	"CA"	<input type="checkbox"/>	<Remove>

At the bottom of the window, there are buttons for '< Add >', '< Clear >', '< See SQL >', '< Do Query >', and '< Or >'.

If it looks different just make the appropriate corrections before continuing.

Now choose **Do Query**. Since we changed the query, the old report form we created doesn't apply anymore so choose **Yes** to overwrite it.

This is the plain vanilla quick report that RQBE produces. When we continue we'll fancy it up again while also grouping and totaling the information by company, like in the report on the previous page.

Close the report window, then use the **File** menu to open the MYQUERY.FRX report again. (Remember to set the **Type** popup to **Report**.)

# Grouping and Subtotaling in RQBE

Modify the header the way you did before. This time move the page footer information, including the page number, up into the header. Make it look like this:

R: 0 C: 15	Move	Page Header
PgHead	DATE()	
PgHead		
PgHead	Company	Phone
PgHead	Contact	Id
Detail	company	phone
	contact	id
PgFoot		
PgFoot		
PgFoot		
PgFoot		

Use the **File** menu to **Save** the report. We'll run this report to see what it looks like, instead of using Page Preview. Click on the RQBE window in the background (or select **RQBE – MYQUERY** on the **Window** menu) to bring that window to the front. Choose **Do Query**.

This report would look better if the information were grouped by company and totaled. Choose **Done** then move to the Report Writer window (use the **Window** menu or click on any part of the Report Writer window, which is in the background).

If **Data Grouping...** is dimmed so you can't select it, unselect all objects in the Report Writer

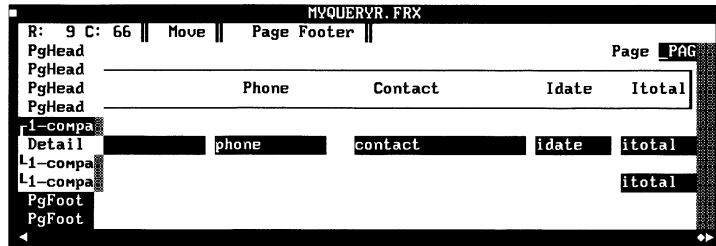
Choose **Data Grouping...** from the **Report** menu, then choose **Add** in the Group dialog. To specify how you want your information grouped, enter **Company** in the text box, then choose **OK**. Choose **OK** again to return to the Report Writer.

FoxPro added group header and footer bands. These work like page headers and footers, except that the information in them is repeated for each group of data instead of each page.

R: 0 C: 15	Move	Page Header
PgHead	DATE()	
PgHead		
PgHead	Company	Phone
PgHead	Contact	Id
1-compa	company	phone
Detail	company	phone
	contact	id
1-compa		
PgFoot		
PgFoot		
PgFoot		
PgFoot		

We need some room to include a subtotal by company, so position the cursor in the Report Layout window on the same line as the group footer label, then select **Add Line** from the **Report** menu.

Scroll right and select the ITOTAL field, then choose **Copy** from the **Edit** menu. Move the cursor down to the bottom line of the group footer, then choose **Paste** from the **Edit** menu.

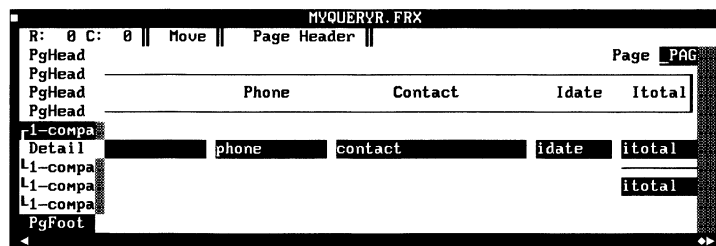


To double-click on a field with the keyboard, position the cursor on the field, then press the Spacebar twice, quickly. If the field is highlighted already, press the Spacebar once to un-select it, then twice more.

To make a line, create a box, then shrink it.

We want the copied field to sum the amounts for each company, so we need to create a “computed field.” Double-click on the copied field, then check the **Calculate...** box in the dialog that appears. FoxPro can perform a lot of different computations. We want a total, so select the **Sum** radio button, then choose **OK**. Choose **OK** again to return to the Report Writer window.

We’ll add a line above the summed field. Position the cursor above the “i” in the bottom ITOTAL field, then select **Box** from the **Report** menu. Shrink the box that appears until it becomes a line. Do so by moving the bottom right corner up and over to the right. Press Enter or release the mouse button when the line is the right length.



Save the report using the **File** menu, then bring the RQBE window to the front while leaving the report window open in the background.

Choose **Do Query** to see what the report looks like now. Scroll right to see the last column.

## One Last Pass with the Report Writer

System File Edit Database Record Program Window Run RQBE				
03/28/92				Page
Company	Phone	Contact	Idate	Itota
Atec Data Service	408/246-5353	Randy Keji	05/17/90	2721.
Atec Data Service	408/246-5353	Randy Keji	05/30/90	163.
Atec Data Service	408/246-5353	Randy Keji	05/26/90	441.
Atec Data Service	408/246-5353	Randy Keji	05/31/90	762.
Atec Data Service	408/246-5353	Randy Keji	05/28/90	744.
				4833.
Automated Mayo Miley	714/540-6062	Bill Hopkins	05/08/90	2336.
				2336.
Azimuth Corp	619/271-8518	Al Reetz	05/08/90	2047.
				2047.
Belmar Tronixs Computer	213/452-9369	Andy Rigney	05/10/90	2353.
				2353.
<< Done >> < More > Column: 0				

Not bad. But it would look better if there were more space between the company data. We can fix that easily.

Choose **Done** to return to the RQBE window, then bring the Report Writer window forward. Move the cursor to the bottom line of the group footer, then add a blank line. Move the copied ITOTAL field up one line.

MVQUERV.R.FRX				
R: 0 C: 0	Move	Page Header	Page PAG	
PgHead				
PgHead				
PgHead				
PgHead				
1-compa				
Detail	phone	contact	idate	itotal
1-compa				
1-compa				itotal
1-compa				
PgFoot				

Now scroll to the left and double-click on the COMPANY field. To prevent company names from repeating, check the **Suppress...** box then select the **On** button and choose **OK**. Choose **OK** again to return to the Report Writer.

Suppress repeated information in the PHONE and CONTACT fields the same way. Then choose **Save** to save the report. Close the Report Writer window.

In the RQBE window, choose **Do Query** to see the final report.

03/20/92			Page	1
Company	Phone	Contact	Idate	Itotal
Atec Data Service	408/246-5353	Randy Keji	05/17/90	2721.19
			05/30/90	163.72
			05/26/90	441.91
			05/31/90	762.56
			05/28/90	744.49
				4833.87
Automated Mayo Miley	714/540-6062	Bill Hopkins	05/08/90	2336.34
				2336.34
Azimuth Corp	619/271-8518	Al Reetz	05/08/90	2047.08
				2047.08

Much nicer, isn't it?

There's more you can do with RQBE and reports, but we've already covered a lot. For more practice with RQBE and reports, see the FoxPro *Getting Started* manual. For information about specific features and options, see the FoxPro *User's Guide*.

Close everything and return to the FoxPro desktop.

RQBE gives you the power to manage your data. You can select, sort, combine and do a lot of things to turn your data into useful information quickly and easily.

And with just one more thing, you'll be ready to create your own custom systems.

We'll use the application generator to show you how to create a screen to view and edit your data. Without doing any programming at all.

## Data Editing and Entry—Your First Application

To select a group of objects, place the mouse pointer to the side and above or below the group of objects, then drag until the dotted line (marquee) covers the group. Release the mouse button. Now you can drag the group as a single unit, or delete it by pressing the Delete key.

To select multiple objects, move the cursor to the first object, then press the Spacebar to select it. Hold the Shift key down while moving to the next object. Press Shift+Spacebar to select each additional object. Move the group using the arrow keys, then press Enter.

To enter and edit data, you need a screen. The Application Generator will create a preliminary one.

Choose **Application...** from the **Run** menu, then choose **New**. A screen needs a database to work with, so choose **File List**, then select and open CUSTOMER.DBF from the TUTORIAL subdirectory.

The application generator suggests a name for the new screen in the bottom half of the dialog, Step 2, so just choose **Create** to accept it. To see the screen, choose **Modify**. This quick screen contains all the fields in the database, labeled with the field names.

A screenshot of a screen titled "CUSTOMER". It displays a list of fields and their corresponding numbers, arranged in two columns. The fields are: Cno (1: cn), Company (2: company.....), Contact (3: contact.....), Address (4: address.....), City (5: city.....), State (6: ), Zip (7: zi), Phone (8: phone....), Ono (9: ), Ytdpurch (10: ytdp), Lat (11: lat), and Long (12: long).

Fields and text in a screen are objects that can be moved or deleted, just like in the Report Writer. For a phone list, all we need is the company, contact and phone fields and the corresponding text. Delete everything else now. (Use the instructions next to the screen above.) Delete the "Company" text object, too, then position the remaining fields as shown in the screen below. (Careful — objects can be stacked on top of each other.) Now resize the screen.

To resize the screen with the mouse, drag the lower right corner where you want it. With the keyboard, use **Screen Layout...** from the **Screen** menu.

A screenshot of a screen titled "CUSTOMER". It displays a modified layout with only the company, contact, and phone fields. The fields are arranged as follows: "1: company....." is at the top. Below it, "Contact 2: contact....." and "Phone 3: phone...." are displayed side-by-side.

Close the Screen Builder now, then choose **Yes** when asked if you want to save it.

This screen needs a program to be useful. FoxPro will write the program for us. Choose **Generate** then save the screen under the suggested name (CUSTOMER.APP). When FoxPro is done, press any key to see your instant application.

CUSTOMER

1st Software Systems Ltd

Contact Rance Sivoren Phone 713/723-1288

< Top > < Prior > < Next > < Bottom > < Search > < Quit >

You can look at the records or change the data in them. The application generator also added a control panel so that you can move around easily in the database. Choose **Next**, **Prior** and **Bottom** to see a few records (including the last one) in the file.

Search for:

93

in: CNO

« OK »

<Cancel>

Choose **Search** then enter 93 in the text box. A dialog explains that a quick search didn't find a match. Choose **Search** in this dialog for a more thorough search. FoxPro finds the record of the customer with this ID (using the CNO field).

Choose **Search** again and open the popup. By selecting a different field name, you can search for different things. Try the contact field and enter Jim as the contact. Case doesn't matter here, so you can enter jim, jIm, or whatever, as long as you spell it right.

Choose **Quit** when you're done. The next time you want to use this application, select **Application...** on the **Run** menu, then select CUSTOMER.APP. It's that easy!

There's a whole lot more you can do with screens and applications. See the FoxPro *Getting Started* manual for more practice. Soon you'll be ready for anything when it comes to managing your information.



## Where Do You Go From Here?

---

You might want to begin with FoxPro *Getting Started* and go through the whole manual. You've already touched on some of the features covered in the tutorial, but you can get more practice and spend time learning about other features.

We haven't covered mailing labels, but the Label Designer chapter in the FoxPro *User's Guide* provides a quick explanation of everything you need to know about creating labels. If you need to create custom labels, choose **Label...** from the **Run** menu, then choose **New** and follow the instructions in the manual.

After that, you might want to dip into the Introduction to FoxPro Power Tools chapter of the FoxPro *User's Guide*. You've already worked with the Screen Builder, RQBE and the Report Writer, so you'll be covering some familiar topics.

Thanks for choosing Microsoft FoxPro.



### 3 Using Files From Other Platforms

---

FoxPro version 2.5 allows you to create and maintain applications that run on multiple platforms, such as Windows and MS-DOS. An application that can run on multiple platforms is a *cross platform application*.

There are several different approaches to writing cross-platform applications in FoxPro, including:

- An approach that automatically runs your MS-DOS applications in Windows – no changes required
- An approach that automatically takes your application from one platform and transports it to another – no coding required
- An approach that allows you to transport your application from one environment to another while maintaining separate interfaces for your application on each platform and maintaining cross platform code compatibility

When you develop applications in FoxPro, you can choose to develop in a character mode environment, such as MS-DOS or UNIX, or in a graphical environment, such as Microsoft Windows or Macintosh. There are advantages to developing on each platform type, but for the most part it's a matter of personal preference.

Because it's easy to transport your applications from platform to platform in FoxPro, you can pick the development platform of your choice. For example, when two people develop different sections of one application, one person may prefer to develop in FoxPro for MS-DOS and the second person may prefer FoxPro for Windows. With FoxPro's cross platform capabilities, it is easy to satisfy both developers by letting them develop on their preferred platforms. Then, you can transport the files to one platform and combine them into an application.

With cross platform applications, you can run an application on multiple platforms and share data with full record locking and other multi-user capabilities.

This chapter describes the methods for developing cross platform applications so that you can choose the best one for your situation. It also explains approaches to designing and maintaining cross platform applications.

The following topics are covered in this chapter:

- Running MS-DOS Applications in Windows
- Maintaining Your Cross Platform Files
- Running Windows Applications in MS-DOS
- The Transporter
- Choosing a Development Platform

## **Cross Platform Glossary**

---

These cross platform terms are used throughout this chapter:

- **Cross Platform Applications** — Applications that can run on multiple platforms, such as MS-DOS, Windows, and so on.
- **Cross Platform Transporter** — A FoxPro program that takes screen, report and label objects created on one platform and duplicates them on another platform.
- **Conversion** — Upgrading a file to the highest version available.
- **Application** — Compiled code that you run.
- **Platform Bracketed Code** — Source code that is divided into sections based on the platform.
- **Object code** — Compiled FoxPro code (.FXPs, .EXEs, and so on).
- **Source code** — Code that you enter or FoxPro generates (.SPRs, .MPRs, and so on). Source code needs to be compiled to run.
- **Controls** — Objects in screens that control your application. These include push buttons, radio button, check boxes, and so on.

## Running MS-DOS Applications in Windows

---

When you decide to run a character mode MS-DOS application in the Windows graphical environment, you can choose one of these three approaches:

1. Run an MS-DOS application as is – no changes required.
2. Use the cross platform Transporter to automatically convert your MS-DOS screens, reports and labels to the graphical Windows format without affecting the interface in MS-DOS – no code changes.
3. Build on the files that were transported in Approach 2 by adding Windows features to your application.

Approaches 2 and 3 add Windows information to the existing MS-DOS application file in a way that allows you to run the application independently on each platform. When you run the application again in MS-DOS, it runs exactly as it did before it was transported.



FoxPro 2.0 screen, report, label, menu and project files are not compatible with FoxPro 2.5 format. When you first open one of these types of files, the file is converted to FoxPro 2.5 format with your permission. Compiled programs (.APPs, .FXPs, and so on) are also not compatible. This section explains how to run each of these file types.

The LASER application, found in C:\FOXPRO25\SAMPLE\LASER, is the example application used throughout this chapter. We've provided several versions of the LASER application that represent the different stages of cross platform development. The applications are located in the following directories:

- C:\FOXPRO25\SAMPLE\LASER\DOSONLY
- C:\FOXPRO25\SAMPLE\LASER\TRANSPRT
- C:\FOXPRO25\SAMPLE\LASER\CRSSPLAT

The figure below shows the application running in MS-DOS. This version of the LASER sample application is found in the C:\FOXPRO25\SAMPLE\LASER\DOSONLY directory. The application name is LASER.APP.

S E D n Laserdisk Library					
Title	2001, A Space Odyssey		Len	149	46
Comments	This release contains lots of "making-of" information. In addition, it is a wonderful transfer.			▲ < Top >	
				◆ < Previous >	
				< Next >	
				< Bottom >	
				< Add >	
Sides	4	Catalog#	CC1160L	Critics	< Browse >
Price	124.95	V.Quality	9		< Report >
Year	1968	Acquired	02/17/89		< Quit >
Rating	PG-13	Category	Science Fict	Studio	Home Vision
<input checked="" type="checkbox"/> Digital transfer <input type="checkbox"/> Closed captioned <input type="checkbox"/> Subtitled <input type="checkbox"/> Commentary <input checked="" type="checkbox"/> Digital audio <input checked="" type="checkbox"/> Letterboxed <input type="checkbox"/> Dubbed <input checked="" type="checkbox"/> Supplements <input checked="" type="checkbox"/> Stereo <input type="checkbox"/> Suitable for kids <input type="checkbox"/> Silent <input checked="" type="checkbox"/> CX encoded <input checked="" type="checkbox"/> Surround sound <input checked="" type="checkbox"/> CAU format <input type="checkbox"/> Black and white					
Finder	2001, A Space Odyssey			Order	TITLE

**Laser Application Running in MS-DOS**  
**C:\FOXPRO25\SAMPLE\LASER\DOSONLY\LASER.APP**



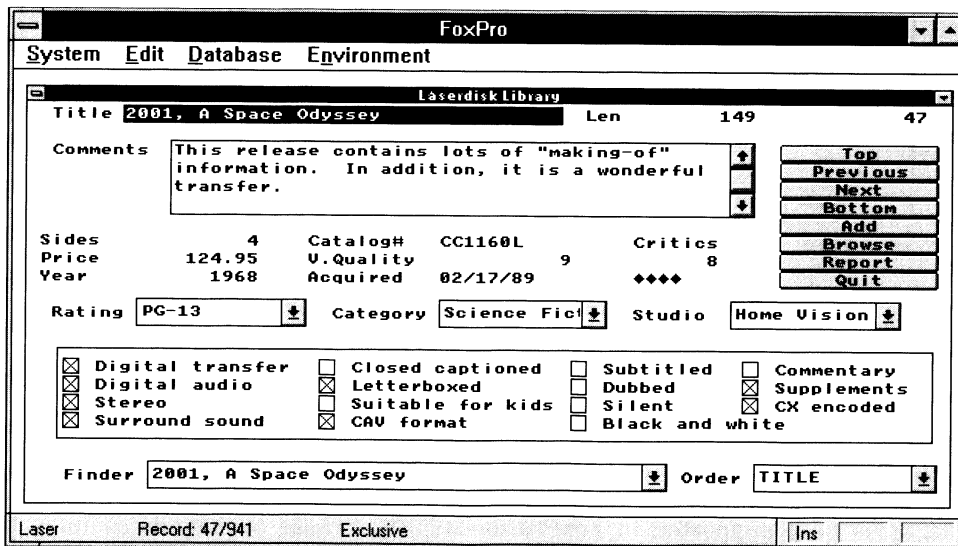
FoxPro 2.5 for MS-DOS and FoxPro 2.5 for Windows create compatible application (.APP and .FXP) files. An application compiled on one platform runs on the other. However, some commands supported in FoxPro for Windows are not supported in FoxPro for MS-DOS. These commands compile but generate runtime errors in MS-DOS.

## Approach 1: Running an MS-DOS Application As Is

If most of your users are running in MS-DOS but you have some users running in Windows, this approach is probably the best solution for you because you can run your MS-DOS applications in Windows without changing anything. Character mode FoxPro applications can be run in the graphical Windows environments without changing code or rearranging objects.

To run an MS-DOS application in Windows, you must first determine which of the following file types you are going to run, then read the appropriate steps:

- Application files (.APP) created using the Project Manager in FoxPro 2.5 for MS-DOS
- FoxPro 2.0 for MS-DOS compiled code
- Program files (.PRG, .SPR, etc.) created in FoxPro for MS-DOS versions 2.0 and 2.5
- Executable files (.EXE) created with the FoxPro for MS-DOS Distribution Kit for versions 2.0 and 2.5
- Applications created in other Xbase products



**MS-DOS Version of Laser Application Running in Windows**  
**C:\FOXPRO25\SAMPLE\LASER\DOSONLY\LASER.APP**



### Running FoxPro 2.5 for MS-DOS Applications

To run a FoxPro 2.5 for MS-DOS application (.APP) in FoxPro for Windows:

1. Choose **Do...** from the **Program** menu.
2. Select the MS-DOS application file and choose **Do**.

### Running FoxPro 2.0 for MS-DOS Applications

Before you run a FoxPro 2.0 for MS-DOS application (.APP) in FoxPro for Windows, you must rebuild it in FoxPro for Windows. You must rebuild because the object code format in FoxPro 2.5 has changed (the object code in FoxPro 2.5 for Windows and FoxPro 2.5 for MS-DOS is identical).

To rebuild the FoxPro 2.0 for MS-DOS application in FoxPro for Windows:

1. Choose **Open...** from the **File** menu then choose the desired project file and choose **Open**. A dialog appears asking permission to convert the 2.0 project to 2.5 format.



Once you convert a 2.0 file you will not be able to open it in FoxPro 2.0 for MS-DOS again. You can, however, open it in FoxPro 2.5 for MS-DOS. If you don't want to convert the project, choose **Cancel** from the dialog.

2. Choose **Yes**. This converts only your project to version 2.5 format. You may be asked permission to relocate the project home directory.
3. Choose the **Build...** push button.
4. Select **Build Application** and choose **OK**.
5. Choose **Yes** to save changes to the project file.
6. Choose **Build** to accept the same file name as the original application.
7. Choose **Yes** to overwrite the existing file.

To run the converted application:

1. Choose **Do...** from the **Program** menu.
2. Select the MS-DOS application file and choose **DO**.

### Running FoxPro for MS-DOS Program Files

It is easy to run a FoxPro for MS-DOS program file in FoxPro for Windows. You don't even need to compile the source code. Just run the program and FoxPro automatically compiles the source code:

1. Choose **Do...** from the **Program** menu.
2. Select the MS-DOS program file and choose **Do**.

### Running FoxPro for MS-DOS Executable Files

FoxPro for MS-DOS executable files from versions 2.0 and 2.5 must be rebuilt and turned into an application (.APP) file before you run them in FoxPro for Windows:

1. Choose **Open...** from the **File** menu to open the corresponding project file. If this project file is in FoxPro 2.0 for MS-DOS format, a dialog appears asking permission to convert the 2.0 project to 2.5 format.



Once you convert a FoxPro 2.0 for MS-DOS file, you will not be able to open it in FoxPro 2.0 for MS-DOS again. You can, however, open it in FoxPro 2.5 for MS-DOS. If you don't want to convert the project, choose **Cancel** from the dialog.

2. Choose **Yes**. This converts only your project to version 2.5 format. You may be asked permission to relocate the project home directory.
3. Choose the **Build...** push button.
4. Select **Build Application** and choose **OK**.

5. Choose **Yes** to save changes to the project file.
6. Choose **Build** to accept the same file name as the original application.
7. Choose **Yes** to overwrite the existing file.

### Running Other Xbase Applications

FoxPro can run other Xbase applications. Some applications will run better than others. In general, the more compatible your code is with FoxPro, the better your application will run. You must have the source code from these Xbase applications so that you can use it to create a project and build a new application. FoxPro can help to identify errors by using the Project Manager and the error log.

To run an Xbase application:

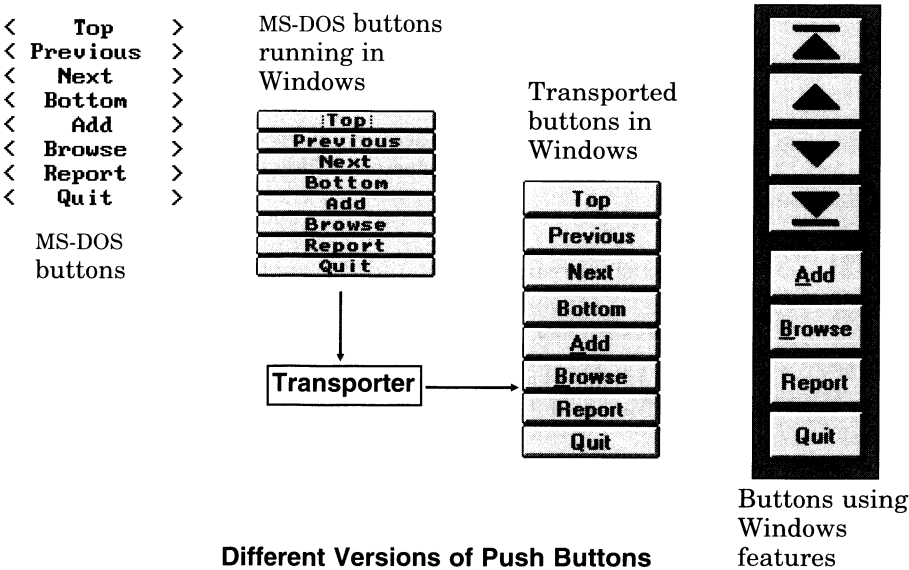
1. Choose **New...** from the **File** menu.
2. Choose **Project** then choose **New**.
3. Add the main program file from the Xbase application. The main program file is used to start the application.
4. Choose the **Build...** push button in the Project window.
5. Select **Build Application**, then choose **Build**. FoxPro automatically pulls all the associated files into the project.
6. Read the error log file (.ERR) to see the compilation errors.
7. Fix the errors and rebuild the application.
8. Run the application.

For additional information, refer to the Compatibility topic in the help file, and the Project Manager chapter in the FoxPro *User's Guide*.

How FoxPro for Windows Runs MS-DOS Applications and Files

There are several differences between character mode (MS-DOS and UNIX) and graphical (Windows and Macintosh) environments. FoxPro takes care of these differences for you behind the scenes:

- **Fonts** — MS-DOS uses a mono-spaced font based on the OEM (Original Equipment Manufacturer) character set and Windows uses an ANSI character set with fonts that are often proportionally spaced. FoxPro for Windows includes a special font, FoxFont, which mimics the MS-DOS OEM character set. FoxFont is automatically used when an MS-DOS application is run in FoxPro for Windows.
- **One line high controls** — FoxPro for MS-DOS controls do not match the size of FoxPro for Windows controls. In many cases, MS-DOS controls are one line high. When an MS-DOS application is run in FoxPro for Windows, a special one line high control is used to match the one line high control in MS-DOS.



- **Half height title bars** — All user-defined windows created in FoxPro for Windows are given a title bar if they include any of the following clauses: TITLE, FLOAT, ZOOM, MINIMIZE or CLOSE. The same is not true of FoxPro for MS-DOS user-defined windows. To reduce the visual differences between applications in the two environments, a half height title bar is used for user-defined windows unless the SYSTEM key word is included or the windows are created with the FONT clause.
- **Box positioning** — Boxes are drawn using the box drawing characters in the FoxFont character set instead of the Windows box drawing routines.
- **Centered popup labels** — Popups in FoxPro for MS-DOS are three lines high and in FoxPro for Windows are approximately one-and-one-half lines high. FoxPro for Windows automatically centers the popup in the three lines when you run a FoxPro for MS-DOS application.

## Approach 2: Transporting MS-DOS Applications

Approach 2 involves transporting each screen, report and label file to the Windows platform. The Transporter handles the details for you and produces a fully functional Windows application. The Transporter does its best to place objects properly. After all files have been transported, rebuild the project and run the application.

When FoxPro transports a screen, report or label file from one platform to another, it creates a duplicate object for every object in the file then modifies the duplicate object for use on the new platform. This does not involve any code changes. You can use the Transporter to create a Windows application, but you'll probably want to do some fine tuning to make the most of the new environment.

Transporting is designed for applications developed with the FoxPro power tools — Screen Builder, Report Writer and Label Designer.

### Transporting MS-DOS Applications Created From a Project

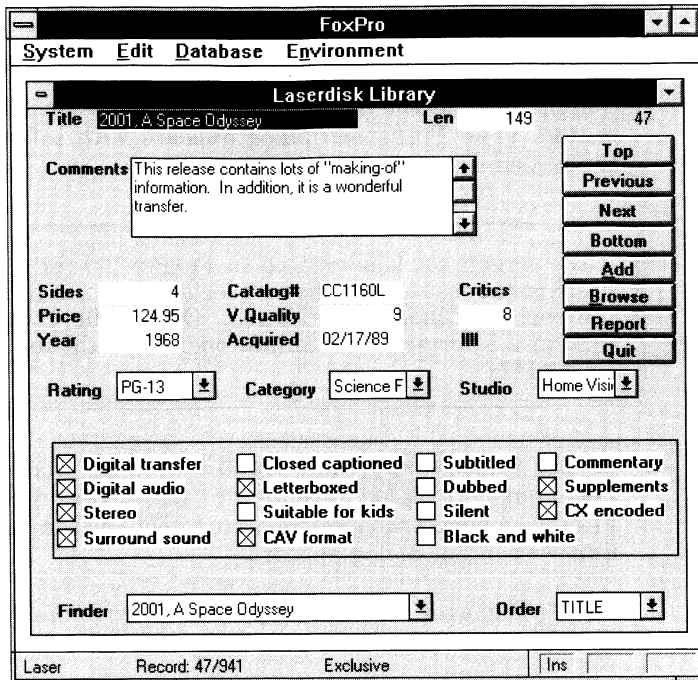
To rebuild the application's project file:

1. Choose **Open...** from the **File** menu then choose the desired project file and choose **Open**. If the project was created in FoxPro 2.0 for MS-DOS, you will be asked whether you want to convert the 2.0 project file to a 2.5 format.



Once you convert a FoxPro 2.0 for MS-DOS file, you will not be able to open it in FoxPro 2.0 for MS-DOS again. You can, however, open it in FoxPro 2.5 for MS-DOS. If you don't want to convert the project file, choose **Cancel** from the dialog.

2. Choose **Yes** to convert your project file to version 2.5 format.



### Laser Application Transported to Windows C:\FOXPRO25\SAMPLE\LASER\TRANSPRT\LASER.APP

#### Transporting Screens, Reports and Labels in a Project

Once you've rebuilt the project, you can begin to transport the individual files contained in the project to the Windows platform. You may want to transport one file at a time then run the application, or you may want to transport several or all of the files and then run the application. Any files you choose not to transport will use FoxFont and one line high control objects.

### Transporting Screens

To transport a screen set in a project:

1. Select a screen set in the Project window then choose the **Edit** button. The Transport dialog appears with information about the screen.



If the screen set was created in FoxPro 2.0 for MS-DOS, the Transporter automatically converts it to 2.5 format when you choose **Transport and Open**. Once a file is converted to version 2.5 format, you cannot open it again in FoxPro 2.0 for MS-DOS.

2. Choose the **Transport and Open** button to process the screen. The Transport dialog appears for each screen in the screen set. After you have transported each screen in the screen set, the Edit Screen Set dialog appears.

The Transporter creates a duplicate object for each object in the MS-DOS screen. Each object includes the following information:

- Platform identification
- A unique ID for each object that is the same as the unique ID for the object's MS-DOS counterpart
- A timestamp for each object

For more information about the Transport dialog and the transporting process, refer to the Transporter section in this chapter.

3. Choose **Edit** to see what your screen looks like or choose **OK** to return to the Project window.



### Transporting Reports and Labels

FoxPro for Windows can print FoxPro for MS-DOS character mode reports and labels. Unless you want to use additional fonts and graphical objects in your reports and labels, there is no reason to transport your FoxPro for MS-DOS reports and labels to FoxPro for Windows.

If you want to use fonts and graphical objects in your reports and labels, follow these steps:

1. Select a report or label file in the Project window then choose **Edit**. The Transport dialog appears with information about the report or labels.



If the report or labels were created in FoxPro 2.0 for MS-DOS, FoxPro automatically converts it to version 2.5 when you choose **Transport and Open**. Once a file is converted to version 2.5, you cannot open it again in FoxPro 2.0.

2. Choose **Transport and Open**.
3. The report or label file opens in the Report Layout window. If necessary, you can make some adjustments to your report.
4. Close the Report Layout window.

### Transporting Menus

Menus do not require transporting because they do not contain information specific to one platform. You can open a FoxPro 2.5 for MS-DOS menu in FoxPro for Windows, make changes, and see them immediately when you return to FoxPro for MS-DOS. FoxPro 2.0 for MS-DOS menus must be converted to run in FoxPro 2.5.

To convert a FoxPro 2.0 for MS-DOS menu file:

1. Select a menu in the Project window. FoxPro asks whether you want to convert the menu to version 2.5 format. If you do convert it, you cannot open it again in FoxPro version 2.0.
2. Choose **Yes**. The menu opens in the Menu Design window.
3. Close the Menu Design window.

## Rebuilding Your Application

You must rebuild your application if you change any files associated with it.

To rebuild an application:

1. Choose the **Build...** push button in the Project window.
2. Select **Build Application** and choose **OK**.
3. Choose **Yes** to save changes to the project.
4. Choose **Build** to accept the same file name as the original application. Then, choose **Yes** to overwrite the the existing file.

With Approach 2, transporting an application requires very little work. FoxPro automatically transports your files without changing any code. Your application now takes advantage of fonts and other Windows features.

Once you see the application running, you may want to continue to Approach 3 to take full advantage of Windows features.

When you edit your files, you can move objects around on the screen. Object positions are stored separately for each platform so an adjustment on one platform does not affect the other.

Each object in a file has a timestamp for each platform. When you edit an object in Windows, the Windows timestamp is updated for that object. These timestamps are used when you transport to another platform. For example, if you update a Windows object in a file and then change to FoxPro for MS-DOS and update the project, FoxPro realizes that there has been a change to a Windows object and asks if you would like to update the corresponding MS-DOS object. Timestamps also help FoxPro identify new objects or changes to code snippets.

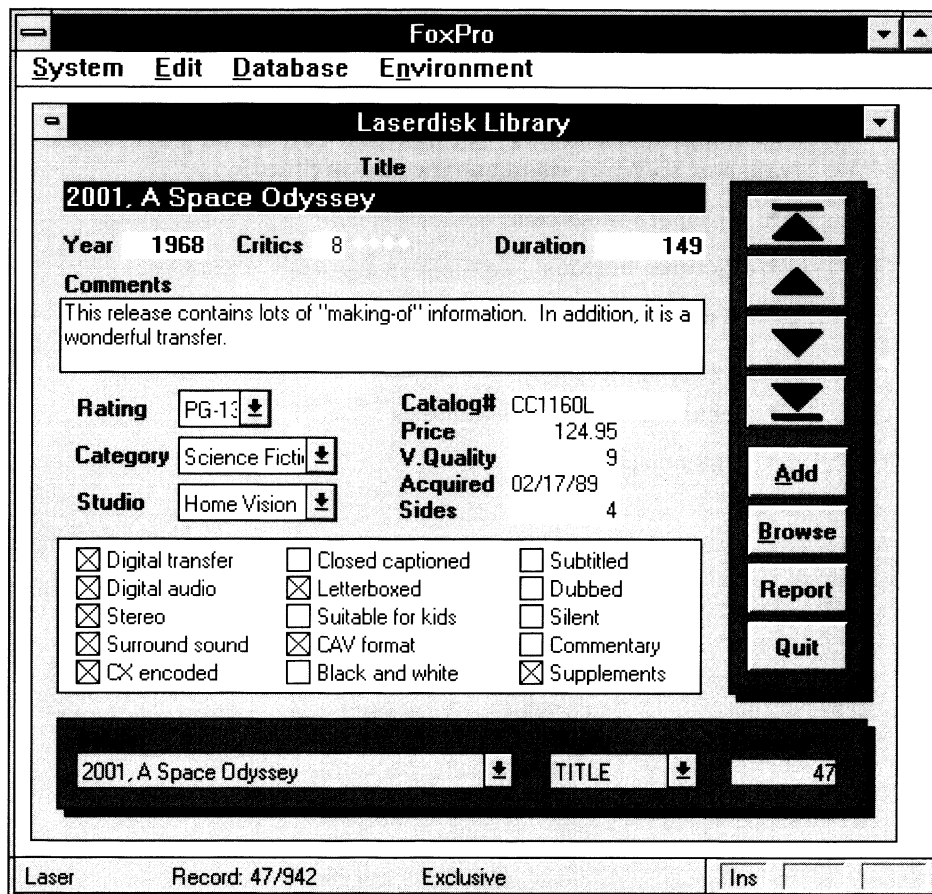
Not all changes to an object update the timestamp. Visual changes such as size, position, and color do not update the timestamp. For a complete list of changes that cause timestamps to update, refer to the Transporting Files topic in the help file.

### **Approach 3: Taking Full Advantage of Windows Features**

We encourage you to use Approach 3 because it allows you to take full advantage of Windows features. For example, you may want to add picture bitmaps, picture buttons and spinners to your application. Or you may just want to change the fonts or colors of your objects.

The following Windows features that can be added to a file you transport to Windows to enhance the application:

- Spinners in screens
- Pictures buttons
- Wallpaper
- Various fonts and font styles
- Icons for minimized windows
- OLE objects



**Laser Application with Full Windows Features**  
**C:\FOXPRO25\SAMPLE\LASER\CRSSPLAT\LASER.APP**

## Maintaining Cross Platform Files

---

Once you've transported your files, you need to consider maintaining your files across platforms. For instance, you must consider how changes to your application in Windows affect your application in MS-DOS, or whether you even want them to affect your application in MS-DOS.

This section discusses considerations for maintaining cross platform applications and assumes that you already have a cross platform application.

### Maintaining Platform Screens

When developing cross platform screens, you should use the Screen Builder for code generation. Then, FoxPro can bracket your cross platform code for you. *Bracketing* means that FoxPro groups platform specific code using CASE statements so that your application knows to run the code corresponding to the current platform.

When you use code snippets in your screens, you need to bracket the code in the snippets yourself when it is specific to one platform. Code bracketing is only necessary when you have Windows-specific items, such as FONT clauses.

When you use the power tools, FoxPro automatically generates the code for your screens. If you have a screen that contains objects from more than one platform, the generated code is bracketed. When you run the application, only the code pertaining to the current platform is executed.

### How Screen Objects are Stored

Screen files are tables of records that store information about each object in a screen. The Transporter creates a duplicate set of records for the new platform. Then, any changes you make to objects on one platform are saved in one set of records so the changes don't impact the other platform. Each object record contains a timestamp field.

### How Objects are Updated

FoxPro uses a timestamp to determine whether an object has been updated. When you change an object, FoxPro may update the object's timestamp. In general, visual changes such as position, color, and size do not update the timestamp. Changes that are made through the object's dialog or the file's layout dialog do up-

date the timestamp. Changes made to the Setup, Cleanup or Read Level clauses also update the timestamp.

FoxPro compares the record timestamps of the current platform to the record timestamps of the platform you are transporting from. If FoxPro finds any differences, then FoxPro figures out which platform record has the very latest timestamp. Then, if the platform that contains the record with the latest timestamp is the platform you are coming from, the Transporter is called. If the latest timestamp is from the current platform records, FoxPro does not call the Transporter.

In the Transport dialog, you can indicate whether you want the change to be reflected on the current platform. You must use the Transporter to update or partially transport the changed file information.



The first time a file is transported, the records that hold environment information (tables, indexes, and so on) are also transported. Environment changes do not have timestamps and do not invoke the Transporter.

### How Objects are Transported

As you begin to expand your applications, you need to make sure that you make changes on all platforms. FoxPro 2.5 knows when changes have been made to a file on a different platform. A timestamp field in the file is updated whenever an object is changed.

When you modify the file after a change has been made to the file on another platform, FoxPro checks the timestamps. If there is a difference in any timestamp, FoxPro brings forward the Transport dialog so that you can do a partial transport. A partial transport allows you to update only the objects that have different timestamps, in other words, only the objects that have changed.

In addition, when you add a new object on another platform, FoxPro can't find information about the object on the current platform. FoxPro calls the Transporter and performs a partial transport to add the new object to the current platform.

The following steps lead you through the process of making a change on one platform then doing a partial transport.

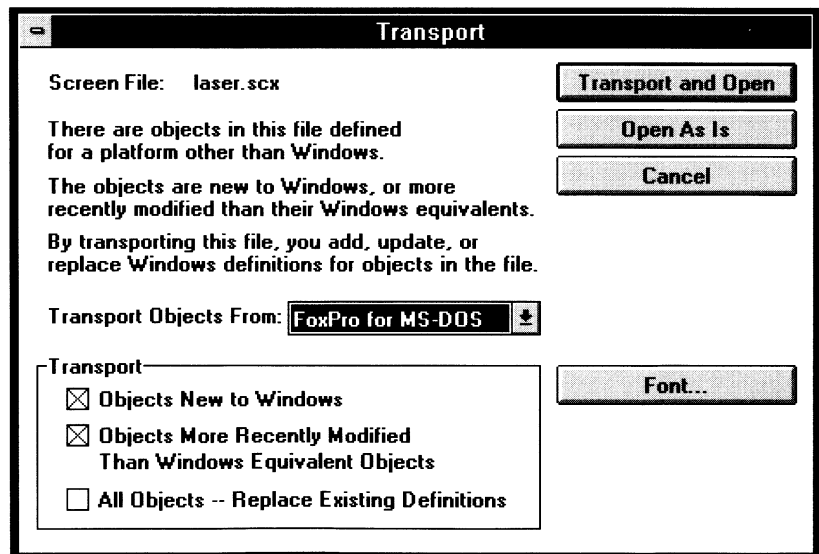
To add an object to a cross platform FoxPro 2.5 for MS-DOS screen:

1. Open the FoxPro 2.5 for MS-DOS screen.
2. Add a new GET field or copy and paste an existing GET field.
3. Save the file.
4. Generate the file.

This file now contains an object for the MS-DOS platform that does not have a duplicate record on the Windows platform. The code that is generated, although bracketed, only shows the object in the MS-DOS code because there isn't a corresponding record for the new object on the Windows platform. To create a record so that the new object appears on the Windows platform, you must modify the file on the Windows platform and do a partial transport.

To modify the cross platform screen file in FoxPro for Windows:

1. Open the screen file. The Transport dialog appears because FoxPro knows that the Windows version of this file is missing the object you just added to the MS-DOS version.



**Transport Dialog for a Partial Transport**

2. Choose **Transport and Open** to add the new object to the Windows version of the screen file. This creates a new record in the screen file so that when you generate the screen, this object is included in the code and appears on the screen.

You can also choose to open the file and not add the object by choosing **Open As Is**.

3. Choose **Generate...** from the **Program** menu. Choose **Generate**.

At this point, everything in the file has been updated. If you run the file, you can see the new object on the screen.

When you modify screen files that contain both MS-DOS and Windows records on one platform, FoxPro invokes the Transporter when you open the file on the other platform.

### How to Bracket Code

When you develop in Windows and use code snippets with Windows-specific code (for example, FONT clauses), bracket your code so that you can transport it to MS-DOS and run it. If you don't bracket your code and you try to run your application in FoxPro for MS-DOS, FoxPro does not give compile errors but it does give runtime errors.

To bracket a code snippet, use a CASE statement like the following:

```
DO CASE

    CASE _DOS

        ***body of MS-DOS code

    CASE _WINDOWS

        ***body of Windows code

ENDCASE
```

Because the platform system memory variable is true only when you are on the corresponding platform, this CASE statement executes the correct code.



Code that is generated by FoxPro from a cross platform file is automatically bracketed. The system memory variables `_DOS` and `_WINDOWS` are used to determine the current platform. They return a true (.T.) value when you are on the corresponding platform.

If you decide not to use the power tools, then you must bracket your own code. Any code that is platform specific needs to be bracketed so that you don't get runtime errors.

You must also bracket code snippets that are added to screens when they contain platform specific code. Look at the LASER application to see how to bracket code snippets.

### How to Generate Exclusive Code

If you do not want your generated code to contain code for both MS-DOS and Windows, you can specify this in the Generate Screen dialog.

To generate exclusive code:

1. Open a cross platform screen.
2. Choose **Generate...** from the **Program** menu to open the Generate Screen dialog.
3. Check the **DOS Objects Only** check box then choose **Generate**.

Now FoxPro generates platform specific code. You can do the same in FoxPro for Windows.

### Maintaining Cross Platform Reports

FoxPro for Windows can print MS-DOS character mode reports. Unless you want to add additional fonts and graphical objects to your reports, there is no reason to transport your FoxPro for MS-DOS reports to FoxPro for Windows.

Report files are tables of records that store information about each object. The Transporter creates a set of records for the new platform that match the records from the other platform. With separate records for each platform in the same file, you can change the file on one platform without changing it on the other.

### Maintaining Cross Platform Labels

Labels can be transported only once. After a label file has been transported, you must make updates on both platforms to maintain cross platform similarities.

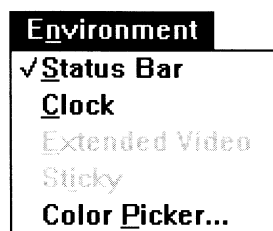
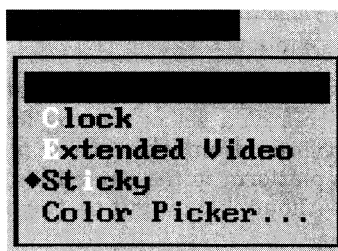
FoxPro for Windows can print MS-DOS character mode labels. Unless you want to add additional fonts and graphical objects to your labels, there is no reason to transport your FoxPro for MS-DOS labels to FoxPro for Windows.

Label files are tables of records that store information about each object. The Transporter creates a set of records for the new platform that match the records from the other platform.

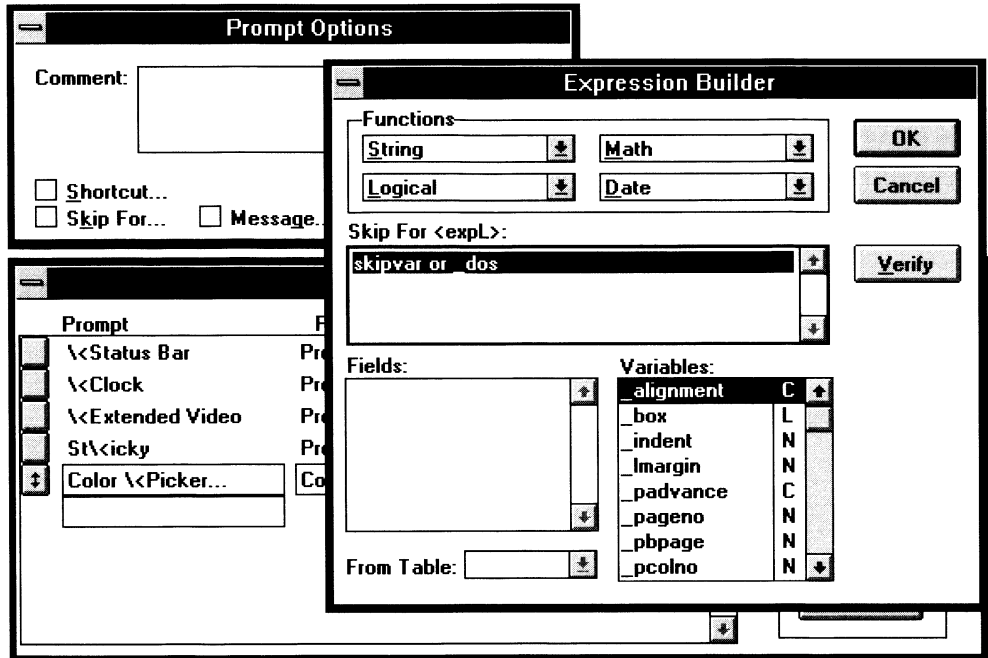
### Maintaining Cross Platform Menus

Menus do not require transporting because they do not contain information specific to one platform. You can open a FoxPro 2.5 for MS-DOS menu in FoxPro for Windows, make changes, and see them immediately when you return to FoxPro for MS-DOS.

In the following example, platform system memory variables are used to enable and disable menu options based on platform. The **Environment** menu has been coded to enable the **Sticky** and **Extended Video** options and disable the **Color Picker...** menu option in the MS-DOS version (in this case, the color picker refers to the Windows Color dialog). For the Windows version, it is the opposite. The **Color Picker...** option is enabled and **Sticky** and **Extended Video** are disabled. The menu file used in this example is located in the C:\FOXPRO25\SAMPLE\LASER\CROSPLAT\MENUS directory.



To create the bracketed code, you need to use a logical expression with the system memory variables. The code example shown in the figure on the next page is from the Expression Builder after you've checked the **Skip For...** check box in the Prompt Options dialog. The expression will be true when the user is in MS-DOS. This code goes with the **Color Picker...** menu option so that when `_DOS` is true (.T.), the option is disabled.



FoxPro for Windows Expression Builder  
Using the System Memory Variable `_DOS`

## **Running Windows Applications in MS-DOS**

---

Running your FoxPro for Windows applications in FoxPro 2.5 for MS-DOS is simple. However, keep the following in mind:

- Not all the code generated in FoxPro for Windows can be run in FoxPro for MS-DOS. For instance, the FONT clause is not understood by FoxPro for MS-DOS. Code like this compiles without any errors but does give runtime errors when you run the program.
- You must transport screens to the MS-DOS platform. This is unlike running an MS-DOS application in Windows where you can run an application without making any changes to the code or using the Transporter.

To transport Windows application files to FoxPro 2.5 for MS-DOS:

1. Open the project file that corresponds to your application.
2. Select a file (screen, report or label) and choose **Edit**. The Transport dialog appears.
3. Choose **Transport and Open** for each file in the project.
4. You can choose to transport one file at a time then modify it, or transport all files before making modifications.

Transporting from the graphical Windows platform to the character mode MS-DOS platform may have some side effects. The Transporter has to choose a good placement for objects. It first has to round off the fractional position and then convert the objects to the right size.

As long as the files are open, the best approach is to take a quick look at the files. Make adjustments as necessary.

5. After all the files have been edited, rebuild the project by choosing **Build....**
6. Select **Rebuild Application** and choose **OK**.
7. Run the application.

This procedure creates a cross platform application. The application code is bracketed.

## Transporter – How Does It Work?

---

The primary goal of the Transporter is to transfer records in project, screen, report and label file tables so that these files function properly on a new platform. This includes information contained in code snippets. To accomplish this, the Transporter adds a duplicate set of records to hold the information for the new platform. This added information allows you to make changes to each platform without affecting the files on the other platform.

The Transporter also does a “best fit” transport to keep applications looking the same no matter which platform they are running on. In some situations, FoxPro cannot transfer objects properly or maintain alignments. The Transporter makes many decisions internally about each object in a file.

When the Transporter creates an identical record for each object in the project, screen, report and label files, the records are appended to the file and marked as either Windows or MS-DOS. When a change is made to a file, FoxPro checks the current platform and makes the change only to the records marked for that platform.

### Transport Dialog

The Transport dialog is used provide specific information to the Transporter about how you want your screens, reports and labels to transport. Controls in the Transport dialog include:

<b>Transport and Open</b>	Choose the <b>Transport and Open</b> button to transport files to the new platform and then open the files in the corresponding design window.
<b>Cancel</b>	Choose <b>Cancel</b> if you decide not to transport the file. If you have a FoxPro 2.0 for MS-DOS file, choosing <b>Cancel</b> means that the file will not be converted.
<b>Transport Objects From</b>	Select the platform that you are transporting your objects from.
<b>Font...</b>	Choose <b>Font...</b> to display the Font dialog so that you can set the default font for the screen, report for label file.

## **Special Transporter Decisions**

All objects in FoxPro for MS-DOS have a counterpart in FoxPro for Windows, so few decisions are needed when transporting objects. FoxPro for Windows objects, however, have many graphical features that aren't supported in MS-DOS and these features require special attention.

For specific information about decisions made by the Transporter, refer to the Transporting Files topic in the help file.

## **Choosing a Development Platform**

---

### **Character Mode MS-DOS Platform**

The character mode environment is considered the lowest common denominator. It is easy to take an application created in character mode and use it in a graphical environment because every object in FoxPro for MS-DOS has a corresponding object in FoxPro for Windows. The reverse is not true. Because a graphical environment offers more possibilities, there are objects in FoxPro for Windows that are not available in FoxPro for MS-DOS.

Once you develop your application in FoxPro for MS-DOS, you can either run your application in the Windows version or you can transport the files so that they become Windows files. These three ways to run your MS-DOS applications in Windows are discussed earlier in this chapter.

When you develop an application in a character mode environment, you can choose the enhancements you want, need or can afford to put into the cross platform application. The decision you make will probably depend on the amount of time you have available, the requirements of your clients, and personal preference.

### **Graphical Windows Platform**

A graphical environment offers many ways to enhance your applications. You can use bitmap pictures for buttons, wallpaper or by themselves (for example, logos). You have complete control of fonts, including size, style and color. You can also add OLE objects, including video and sound. This makes developing in the Windows environment very appealing.

There are a few things to remember when transporting to character mode environments, including what happens to Windows features like fonts, bitmap pictures and OLE objects. In FoxPro 2.5, the code that is generated using the power tools is bracketed into platform specific code. FoxPro executes only the necessary code when you are in Windows or in MS-DOS.

The system memory variables `_DOS` and `_WINDOWS` each return a true (.T.) value when you are on the respective platform. Using a CASE statement with these system memory variables, FoxPro brackets the code so that the correct platform version of the files display on your screen.





## 4 Tables

---

This chapter contains the following tables:

- File Types and Extensions
- System Capacities
- Control Key Shortcuts
- Table Structures
  - .PJX Table (Projects)
  - .SCX Table (Screens)
  - .FRX Table (Reports)
  - .MNX Table (Menus)
  - .LBX Table (Labels)
- Other File Structures
  - Table File Structure (.DBF)
  - Memo and General File Structure (.FPT)
  - Index File Structure (.IDX)
  - Compact Index File Structure (.IDX)
  - Compound Index File Structure (.CDX)
  - FoxPro 2.0 Project File Structure (.PJX)
  - FoxPro 2.0 Screen File Structure (.SCX)
  - FoxPro 2.0 Report File Structure (.FRX)
  - FoxPro 2.0 Label File Structure (.LBX)
  - FoxPro 1.x Report File Structure (.FRX)
  - FoxPro 1.x Label File Structure (.LBX)
  - FoxBASE+ Memo File Structure (.DBT)
  - Macro File Format (.FKY)

## **File Types and Extensions**

---

<b>File Type</b>	<b>Extension</b>
Table	.DBF
Dynamic Link Library (FoxPro for Windows)	.FLL
Memo	.FPT
Memo Backup	.TBK
FoxBASE+ Style Memo	.DBT
Index	.IDX
Compound Index	.CDX
Program	.PRG
Compiled Program	.FXP
Format	.FMT
Compiled Format	.PRX
View	.VUE
Text	.TXT
File Backup	.BAK
Report	.FRX
Report Memo	.FRT
Label	.LBX
Label Memo	.LBT
Screen	.SCX
Screen Memo	.SCT
Generated Screen Program	.SPR
Compiled Screen Program	.SPX

---

File Type		Extension
Menu		.MNX
Menu Memo		.MNT
Generated Menu Program		.MPR
Compiled Menu Program		.MPX
Generated Query Program		.QPR
Compiled Query Program		.QPX
Project		.PJX
Project Memo		.PJT
Generated Application		.APP
Executable Program		.EXE
Compilation Error File		.ERR
Memory Variable Save		.MEM
Macro Files		.FKY
Window File		.WIN
Resource Files	FOXUSER.DBF FOXUSER.FPT FOXUSER.CDX	
Help Files	FOXHELP.DBF FOXHELP.FPT	
Configuration File	CONFIG.FP	
Temporary File		.TMP
FoxDoc Reports		.DOC
FoxDoc Action Diagrams		.ACT
Libraries		.PLB

## System Capacities

System Capacities		
	FoxPro	FoxPro Extended
<b>Database and Index Files</b>		
Maximum # of records per database file	1 billion <sup>1</sup>	1 billion <sup>1</sup>
Maximum # of characters per record	65,500	65,000
Maximum # of fields per record	255	255
Maximum # of databases open at one time	25	225
Maximum # of characters per database field	254	254
Maximum # of characters per index key (.IDX)	100	100
Maximum # of characters per index key (.CDX)	240	240
Maximum # of open index files per database	unlimited <sup>2</sup>	unlimited <sup>2</sup>
Maximum # of open indexes in all work areas	unlimited <sup>2</sup>	unlimited <sup>2</sup>
<b>Field Characteristics</b>		
Maximum size of character fields	254	254
Maximum size of numeric (and float) fields	20	20
Maximum number of characters in field names	10	10
Digits of precision in numeric computations	16	16
<b>Memory Variables and Arrays</b>		
Default # of memory variables	256	256
Maximum # of memory variables	3,600	65,000
Maximum # of arrays	3,600	65,000
Maximum # of elements per array	3,600	65,000
<b>Program and Procedure Files</b>		
Maximum # of lines in source program files	unlimited	unlimited
Maximum size of compiled program modules <sup>3</sup>	64K	64K
Maximum # of procedures per file	unlimited	unlimited

<sup>1</sup>The actual file size (in bytes) cannot exceed 2 gigabytes for single-user or exclusively opened multi-user .DBF files. Shared multi-user .DBF files with no indexes or .IDX indexes cannot exceed 1 gigabyte. Shared multi-user .DBF files with structural .CDX indexes cannot exceed 2 gigabytes.

<sup>2</sup>Limited by memory and available file handles. .CDX files use only one file handle.

<sup>3</sup>A program *module* is one procedure. A program or application can contain an unlimited number of program modules.

System Capacities		
	FoxPro	FoxPro Extended
<b>Program and Procedure Files</b>		
Maximum # of nested DO calls	32	32
Maximum # of READ nesting levels	5	5
Maximum # of structured programming commands	64	64
Maximum # of procedure arguments	24	24
<b>Report Writer Capacities</b>		
Maximum # of objects in a report definition	unlimited	unlimited
Maximum # of lines in a report definition	255	255
Maximum # of grouping levels	20	20
<b>Window Support</b>		
Maximum # of open windows (all types)	unlimited <sup>2</sup>	unlimited <sup>2</sup>
Maximum # of open Browse windows	25	25
<b>Miscellaneous Capacities</b>		
Maximum # of characters per character string	64K	2 gigabytes
Maximum # of characters per command line	2,048	2,048
Maximum # of characters per macro subst. line	2048	2048
Maximum # of open files	99	DOS limit
Maximum keystrokes in keyboard macro	1024	1024
Maximum fields that can be selected by a SQL SELECT statement	255	255
<b>Color Support</b>		
Number of color schemes per color set	24	24
Maximum # of color sets (in FOXUSER file)	unlimited	unlimited
Number of colors per color scheme	10	10
Schemes available for user definition	8	8

## Control Key Shortcuts

---

Key	Action
Ctrl+A	Select All
Ctrl+C	Copy selected items to clipboard
Ctrl+D	Specify a program to execute
Ctrl+E	Replace text and find next occurrence
Ctrl+F	Find specified text
Ctrl+G	Find next occurrence of specified text
Ctrl+K	Locate next occurrence of specified record
Ctrl+M	Resume executing a suspended program
Ctrl+Q	Exit current edit without saving changes; Exit dialogs without taking action
Ctrl+R	Redo a text editing action you just undid
Ctrl+V	Paste item from the clipboard
Ctrl+W	Exit current edit and save any modifications
Ctrl+X	Remove selected items and save on clipboard
Ctrl+Z	Undo a text editing action
Ctrl+F1	Cycle windows
Ctrl+F2	Display Command window
Ctrl+F7	Move frontmost window
Ctrl+F8	Size frontmost window
Ctrl+F9	Minimize frontmost window
Ctrl+F10	Zoom frontmost window
Esc	Exit current edit without saving changes; Exit dialogs without taking action

## Table Structures

---

The tables on the following pages contain the structures of the project, menu, screen, report and label data files. These tables show the fields in the data files and the type of information stored in the fields. The structures of these data files are subject to change.

## **.PJX Table (Projects)**

Fields	Type	Header	Screen Set	Screen	Program	Menu
NAME <sup>1</sup>	M	text	set name	name	name	name
TYPE	C	H	S	S	P	M
TIMESTAMP	N	time stamp	time stamp	time stamp	time stamp	time stamp
OUTFILE	M	location <sup>2</sup>	output file <sup>1</sup>			output file <sup>1</sup>
HOMEDIR	M	homedir	homedir			homedir
SETID	N	highest id	key number	key number		
EXCLUDE	L		exclude?		exclude?	exclude?
MAINPROG	L		main?		main?	main?
ARRANGED	M			arrange info		
SAVECODE	L	saved?	Windows objects only			
DEFNAME	L		defaulted?			
OPENFILES	L		open files?			
CLOSEFILES	L		close files?			
DEFWINDS	L		define winds?			
RELWINDS	L		release winds?			
READCYCLE	L		cycle?			
MULTREADS	L		multiple?			
NOLOCK	L		lock?			
MODAL	L		modal?			
ASSOCWINDS	M		window list			
DEBUG	L	debug?				
ENCRYPT	L	encrypt?				
NOLOGO	L		show logo?			
SCRNORDER	N			order number		
CMNTSTYLE	N	box/asterisk				
OBJREV	N		obj rev		obj rev	obj rev
COMMANDS	M		bitmap		bitmap	bitmap
DEVINFO	M	developer info				
SYMBOLS	M		symbol table		symbol table	symbol table
OBJECT	M		obj code		obj code	obj code
CKVAL	N					

<sup>1</sup>Includes normalized path.



[illegible]

<sup>2</sup>Location of the generated code (<Source>, <Project> or path).

## **.SCX Table (Screens)**

Fields	Type	Screen	Workarea	Index	Relation	Text	Box	Line	Group
PLATFORM	C	ALL	ALL	ALL	ALL	ALL	ALL	M/W	ALL
UNIQUEID	C	ALL	ALL	ALL	ALL	ALL	ALL	M/W	ALL
TIMESTAMP	N								
OBJTYPE	N	ALL(1)	ALL(2)	ALL(3)	ALL(4)	ALL(5)	ALL(7)	M/W(6)	ALL(10)
OBJCODE	N	ALL(vers.)	ALL(1-25)*	ALL(1-25)*	ALL(1-25)*	ALL(0)	ALL(3-6)		D/U(0-1)
NAME	M	ALL(win.)	ALL file .dbf	ALL file .idx					
EXPR	M		ALL set skip	ALL idx exp	ALL rel exp	ALL text			
VPOS	F	ALL				ALL	ALL	M/W	ALL obj no
HPOS	F	ALL				ALL	ALL	M/W	ALL obj ct
HEIGHT	F	ALL				ALL	ALL	M/W	
WIDTH	F	ALL				ALL	ALL	M/W	
STYLE	N	ALL usr/dlg				M/W jmod	M/W radius	M/W horiz	
PICTURE	M								
ORDER	M		ALL indx name						
UNIQUE	L		ALL (= .T.)	ALL					
COMMENT	M		ALL	ALL	ALL	ALL	ALL		
ENVIRON	L	ALL							
BOXCHAR	C						D/U		
FILLCHAR	C						D/U		
TAG	M	ALL title	ALL alias	ALL for exp	ALL into alias				
TAG2	M	D/U footer	ALL tag name	ALL to alias	ALL from alias				
PENRED	N					M/W	M/W	M/W	
PENGREEN	N					M/W	M/W	M/W	
PENBLUE	N					M/W	M/W	M/W	
FILLRED	N	M/W					M/W	M/W	
FILLGREEN	N	M/W					M/W	M/W	
FILLBLUE	N	M/W					M/W	M/W	
PENSIZE	N					M/W	M/W		
PENPAT	N						M/W	M/W	
FILLPAT	N						M/W	M/W	
FONTFACE	M	M/W				M/W			
FONTSTYLE	N	M/W				M/W			
FONTSIZE	N	M/W				M/W			
MODE	N					M/W	M/W	M/W	
RULER	N	M/W ruler							
RULERLINES	N	M/W line							
GRID	L	M/W grid							
GRIDV	N	M/W vert							
GRIDH	N	M/W horiz							

M/W = Macintosh/Windows D/U = MS-DOS/UNIX

\*All(1-225) in 32-bit Extended version

This table is continued on page A-12

List	Textbutn	Radiobutn	Checkbox	Getfield	Texttrgn	Popup	Picture	Spinner	Invisiblebtn
ALL	ALL	ALL	ALL	ALL	ALL	ALL	M/W	M/W	ALL
ALL	ALL	ALL	ALL	ALL	ALL	ALL	M/W	M/W	ALL
ALL(11)	ALL(12)	ALL(13)	ALL(14)	ALL(15)	ALL(15)	ALL(16)	M/W(17)	M/W(22)	ALL(20)
ALL(2)	ALL(1)	ALL(1)	ALL(1)	ALL(0,1)	ALL(1)	ALL(1)		M/W(1)	ALL(1)
ALL v name	ALL v name	ALL v name	ALL v name	ALL v name	ALL v name	ALL v name	M/W fieldname	M/W v name	ALL v name
ALL frm exp				ALL say exp		ALL say exp			
ALL	ALL	ALL	ALL	ALL	ALL	ALL	M/W	M/W	ALL
ALL	ALL	ALL	ALL	ALL	ALL	ALL	M/W	M/W	ALL
ALL	ALL	ALL	ALL	ALL	ALL	ALL	M/W	M/W	ALL
ALL	ALL	ALL	ALL	ALL	ALL	ALL	M/W	M/W	ALL
ALL(0-4)							M/W from type		
	ALL func	ALL func	ALL func	ALL fun/pict	ALL pict	ALL func	M/W pict file	M/W pic	ALL pic
ALL	ALL	ALL	ALL	ALL	ALL	ALL	M/W	M/W	ALL
				ALL type					
								min value	
								max value	
M/W	M/W	M/W	M/W	M/W	M/W	M/W	M/W		
M/W	M/W	M/W	M/W	M/W	M/W	M/W	M/W		
M/W	M/W	M/W	M/W	M/W	M/W	M/W	M/W		
M/W	M/W	M/W	M/W	M/W	M/W	M/W	M/W		
M/W	M/W	M/W	M/W	M/W	M/W	M/W	M/W		
M/W	M/W	M/W	M/W	M/W	M/W	M/W	M/W		
M/W	M/W	M/W	M/W	M/W	M/W	M/W			
M/W	M/W	M/W	M/W	M/W	M/W	M/W			
M/W	M/W	M/W	M/W	M/W	M/W	M/W			
M/W	M/W	M/W	M/W	M/W	M/W	M/W	M/W		
	ALL	ALL							
	ALL	ALL							
	ALL	ALL							
	ALL	ALL							
	ALL	ALL							

M/W = Macintosh/Windows      D/U = MS-DOS/UNIX

This table is continued on page A-13

## **.SCX Table (Continued)**

Fields	Type	Screen	Workarea	Index	Relation	Text	Box	Line	Group
SCHEME	N	D/U				D/U	D/U		
SCHEME2	N	D/U							
COLORPAIR	C					D/U	D/U		
LOTYP	N								
RANGELO	M								
HITYPE	N								
RANGEHI	M								
WHENTYPE	N	ALL exp code							
WHEN	M	ALL							
VALIDTYPE	N	ALL exp code							
VALID	M	ALL							
ERRORTYPE	N								
ERROR	M								
MESSTYPE	N								
MESSAGE	M								
SHOWTYPE	N	ALL exp code							
SHOW	M	ALL							
ACTIVTYPE	N	ALL exp code							
ACTIVATE	M	ALL							
DEACTTYPE	N	ALL exp code							
DEACTIVATE	M	ALL							
PROCTYPE	N	ALL exp code							
PROCCODE	M	ALL							
SETUPTYPE	N	ALL exp code							
SETUPCODE	M	ALL							
FLOAT	L	ALL							
CLOSE	L	ALL							
MINIMIZE	L	ALL							
BORDER	N	ALL						M/W dble	
SHADOW	L	ALL							
CENTER	L	ALL							
REFRESH	L					ALL			
DISABLED	L								
SCROLLBAR	L								
ADDALIAS	L	ALL							
TAB	L								
INITIALVAL	M								
INITIALNUM	N	ALL init obj							
SPACING	F					M/W			
CURPOS	L	M/W							

M/W = Macintosh/Windows    D/U = MS-DOS/UNIX

[illegible]

M/W = Macintosh/Windows      D/U = MS-DOS/UNIX

## .FRX Table (Reports)

Fields	Type	Report	Bandinfo	Picture	Repfield	Text	Box
PLATFORM	C	ALL platform	ALL platform	ALL platform	ALL platform	ALL platform	ALL platform
UNIQUEID	C	ALL uniqueid	ALL uniqueid	ALL uniqueid	ALL uniqueid	ALL uniqueid	ALL uniqueid
TIMESTAMP	N						
OBJTYPE	N	ALL(1)	ALL(9)	ALL(17)	ALL(8)	ALL(5)	ALL(7)
OBJCODE	N	ALL version	ALL TYPE(0-8)	M/W SAY(0)	ALL SAY(0)	ALL SAY(0)	D/U (3-6)
NAME	M			fieldname			
EXPR	M		ALL group expr.		ALL the text	ALL SAY expr.	
VPOS	F	M/W		M/W vpos	ALL vpos	ALL vpos	ALL vpos
HPOS	F	M/W		M/W hpos	ALL hpos	ALL hpos	ALL hpos
HEIGHT	F	ALL page len	ALL height	M/W height	ALL height	ALL height	ALL height
WIDTH	F	ALL pg. width		M/W width	ALL width	ALL width	ALL width
STYLE	M				ALL style	ALL style	
PICTURE	M				ALL fun/pict.		
ORDER	M						
UNIQUE	L						
COMMENT	M				ALL comment	ALL comment	ALL comment
ENVIRON	L	ALL environ?					
BOXCHAR	C						D/U boxchar
FILLCHAR	C				D/U		D/U fillchar
TAG	M						
TAG2	M						
PENRED	N	M/W			M/W	M/W	M/W
PENGREEN	N	M/W			M/W	M/W	M/W
PENBLUE	N				M/W	M/W	M/W
FILLRED	N				M/W		M/W
FILLGREEN	N				M/W		M/W
FILLBLUE	N				M/W		M/W
PENSIZE	N						M/W
PENPAT	N						M/W
FILLPAT	N						M/W
FONTFACE	M				M/W	M/W	
FONTSTYLE	N				M/W	M/W	
FONTSIZE	N				M/W	M/W	
MODE	N			M/W	M/W	M/W	M/W
RULER	N	M/W					
RULERLINES	N	M/W					
GRID	L	M/W					
GRIDV	N	M/W					
GRIDH	N	M/W					
FLOAT	L				ALL float?	ALL float?	ALL float?
STRETCH	L				ALL stretch?		
STRETCHTOP	L	M/W		M/W	M/W		
TOP	L	M/W		M/W	M/W		
BOTTOM	L	M/W		M/W	M/W		
SUPTYPE	N			M/W	M/W		
SUPREST	N			M/W	M/W		
NOREPEAT	L	ALL	D/U		ALL norepeat		
RESETRPT	N				ALL resetrpt		
PAGEBREAK	L		ALL				
COLBREAK	L		W/M				
RESETPAGE	L		ALL				
GENERAL	N			stretch style			
SPACING	N				W/M spacing		

This table is continued on page A-16

[illegible]

\*All(1-225) in 32-bit Extended version

This table is continued on page A-17

## **.FRX Table (continued)**

<b>Fields</b>	<b>Type</b>	<b>Report</b>	<b>Bandinfo</b>	<b>Picture</b>	<b>Repfield</b>	<b>Text</b>	<b>Box</b>
DOUBLE	L			center?			
SWAPHEADER	L		D/U				
SWAPFOOTER	L		D/U				
EJECTBEFOR	L	D/U					
EJECTAFTER	L	D/U					
PLAIN	L	D/U					
SUMMARY	L	D/U					
ADDALIAS	L	ALL					
OFFSET	N	D/U	D/U	fromtype		M/W just	M/W radius
TOPMARGIN	N	D/U			M/W jmod		
BOTMARGIN	N	D/U					
TOTALTYPE	N				ALL total		
RESETTOTAL	N				ALL reset		
RESOID	N						
CURPOS	L	M/W					
SUPALWAYS	L			ALL	ALL	ALL	ALL
SUPOVFLOW	L			ALL	ALL	ALL	ALL
SUPRPOOL	N			ALL	ALL	ALL	ALL
SUPGROUP	N			ALL	ALL	ALL	ALL
SUPVALCHNG	L			ALL	ALL	ALL	ALL
SUPEXPR	M			ALL	ALL	ALL	ALL

M/W = Macintosh/Windows    D/U = MS-DOS/UNIX





## **.MNX Table (Menus)**

---

Fields		Object Types and Field Data		
OBJTYPE	N	MENUSYSTEM (1)	SUBMENU (2)	ITEM (3)
OBJCODE	N	version		
NAME	M		menu name	pad name
PROMPT	M			prompt
COMMAND	M			command
MESSAGE	M	menu pad message	submenu message	item message
PROCTYPE	N			
PROCEDURE	M	global default	menu options	item
SETUPTYPE	N			
SETUP	M	procedure		
CLEANTYPE	N			
CLEANUP	M	procedure		
MARK	C	mark (global)	mark (menu)	mark (item)
KEYNAME	M			key name
KEYLABEL	M			key label
SKIPFOR	M			skip for
NAMECHANGE	L		changed name?	
NUMITEMS	N		number of items	
LEVELNAME	C	menu level name	menu level name	menu level name
ITEMNUM	C	item number	item number	item number
COMMENT	M			comment
LOCATION	N	location of menu		
SCHEME	N		scheme	

## **.LBX Table (Labels)**

---

The structure of label tables (.LBX) is the same as that of report tables (.FRX).

# Table File Structure (.DBF)

A table file is made up of a header record and data records. The header record defines the structure of the table and contains any other information related to the table. It starts at file position zero.

The data records<sup>1</sup> follow the header (in consecutive bytes) and contain the actual text of the fields. The length of a record (in bytes) is determined by summing the defined lengths of all fields. Numbers in this file are represented in reverse bytes.

Table Header Record	
Bytes	Description
00	Type of data file: FoxBASE+®/dBASE III PLUS®, no memo — 0x03 FoxBASE+®/dBASE III PLUS, with memo — 0x83 FoxPro/dBASE IV, no memo — 0x03 FoxPro with memo — 0xF5 dBASE IV with memo — 0x8B
01-03	Last update (YYMMDD)
04-07	Number of records in file
08-09	Position of first data record
10-11	Length of one data record (including delete flag)
12-31	Reserved
32-n	Field subrecords <sup>2</sup>
n + 1	Header record terminator (0x0D)

Field Subrecords <sup>3</sup>	
Bytes	Description
00-10	Field name (maximum of 10 characters — if less than 10 it is padded with null character (0x00))
11	Data Type: C – Character N – Numeric L – Logical M – Memo G – General D – Date F – Float P – Picture
12-15	Displacement of field in record
16	Length of field (in bytes)
17	Number of decimal places
18-32	Reserved

### Notes to Data File Structure

<sup>1</sup>The data in the data file starts at the position indicated in bytes 08-09 of the header record. Data records begin with a delete flag byte. If this byte is an ASCII space (0x20) the record is not deleted; if the first byte is an asterisk (0x2A) the record is deleted. The data from the fields named in the field subrecords follows the delete flag.

<sup>2</sup>The number of fields determines the number of field subrecords. There is one field subrecord for each field in the table.

<sup>3</sup>See the System Capacities table in this appendix for limitations on characters per record, maximum fields, etc.

## **Memo and General File Structure (.FPT)**

Memo files contain one header record and any number of block structures. The header record contains a pointer to the next free block and the size of the block in bytes. The size is determined by the SET BLOCKSIZE command when the file is created. The header record starts at file position zero and occupies 512 bytes.

Following the header record are the blocks that contain a block header and the text of the memo. The table file contains block numbers that are used to reference the memo blocks. The position of the block in the memo file is determined by multiplying the block number by the block size (found in the memo file header record). All memo blocks start at even block boundary addresses. A memo block can occupy more than one consecutive block.

<b>Memo Header Record</b>	
<b>Bytes</b>	<b>Description</b>
00-03	Location of next free block <sup>1</sup>
04-05	Unused
06-07	Block size (bytes per block) <sup>1</sup>
08-511	Unused
<b>Memo Block Header and Memo Text</b>	
00-03	Block signature <sup>1</sup> (indicates type of data in block): a. 0 – picture (picture field type) b. 1 – text (memo field type)
04-07	Length <sup>1</sup> of memo (in bytes)
08-n	Memo text (n = length)

<sup>1</sup>Number represented in left-to-right order in hexadecimal.

## **Index File Structure (.IDX)**

Index files contain one header record and one or many node records. The header record contains information about the root node, the current file size, the length of the key, index options and signature, and printable ASCII representations of the key<sup>1</sup> and FOR expressions. The header record starts at file position zero.

The remaining node records contain an attribute, number of keys present and pointers to nodes on the left and right (on the same level) of the current node. They also contain a group of characters encompassing the key value and either a pointer to a lower level node or an actual table record number. The size of each record that is output to file is 512 bytes.

An example of an ordered tree structure follows the tables.

<b>Index Header Record</b>	
<b>Byte</b>	<b>Description</b>
00-03	Pointer to root node
04-07	Pointer to free node list (-1 if not present)
08-11	Pointer to end of file (file size)
12-13	Length of key
14	Index options (any of the following numeric values or their sums): a. 1 – a unique index b. 8 – index has FOR clause
15	Index signature (for future use)
16-235	Key expression (uncompiled; up to 220 characters) <sup>1,3</sup>
236-455	FOR expression (uncompiled; up to 220 characters ending with null byte)
456-511	Unused

Index Node Record	
Byte	Description
00-01	Node attributes (any of the following numeric values or their sums): a. 0 – index node b. 1 – root node c. 2 – leaf node
02-03	Number of keys present (0, 1 or many)
04-07	Pointer to node directly to left of current node (on same level; -1 if not present)
08-11	Pointer to node directly to right of current node (on same level; -1 if not present)
12-511	Up to 500 characters containing the key value for the length of the key with a four-byte hexadecimal number (stored in normal left-to-right format): If the node is a leaf (attribute = 02 or 03) then the four bytes contain an actual table number in hexadecimal format <sup>2</sup> — else the 4 bytes contain an intra-index pointer. <sup>2</sup> The key/four-byte hexadecimal number combinations will occur the number of times indicated in bytes 02-03.

<sup>1</sup>The type of the key is not stored in the index. It must be determined by the key expression.

<sup>2</sup>Anything other than character strings, numbers used as key values, and the four-byte numbers in the leaf node are represented in reversed bytes (Intel 8086 format).

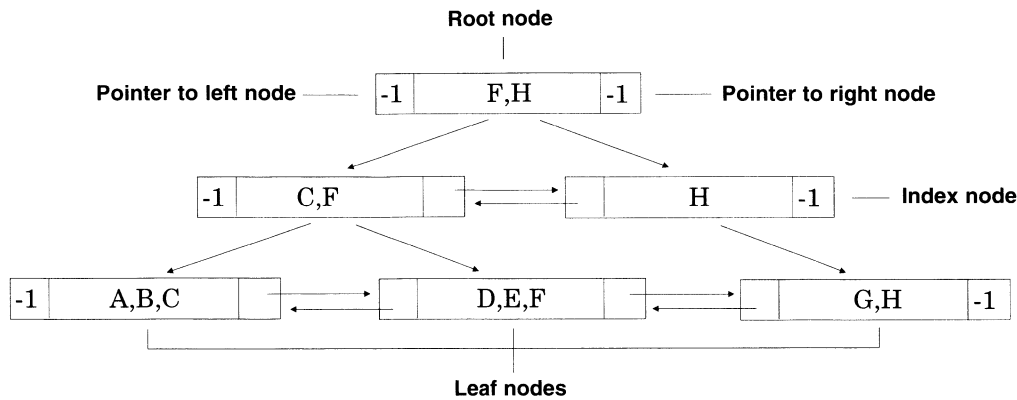
<sup>3</sup>Numbers are a special case when used as a key. They are converted through the following algorithm so they can be sorted using the same ASCII collating sequence as characters:

- Convert the number to IEEE floating point format.
- Swap the order of the bytes from Intel 8086 order to left-to-right order.
- If the number was negative, take the logical complement of the number (swap all 64 bits, 1 to 0 and 0 to 1) else invert only the leftmost bit.



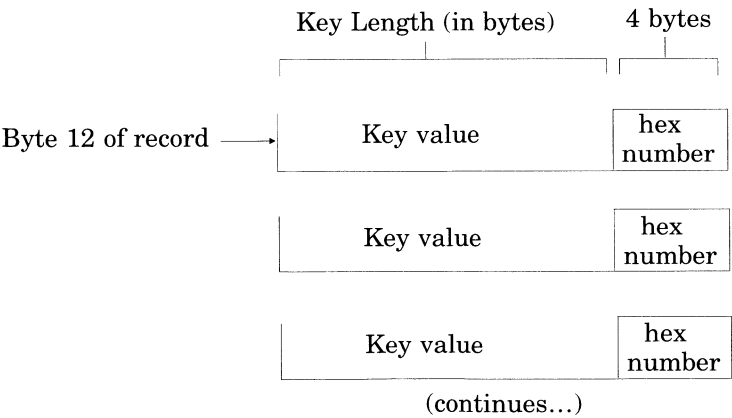
Example of an Ordered Tree Structure

Finding a key in the structure below requires searching a single path between the root and leaf nodes. Nodes at the lowest level are leaf nodes. Because the keys are sorted, all keys in the subtree are less than or equal to the parent node.



In the illustration above, the letters are used as the key values. Each key would also have a four-byte hexadecimal number. The numbers associated with the keys in the *leaf* nodes would be actual table numbers — all keys in other nodes would have intra-index pointers associated with them.

Bytes 12-511 in the index node records could be viewed as follows:



The key value/hexadecimal number combination occurs in bytes 12-511 n times where n is the number of keys present.

## Compact Index File Structure (.IDX)

Compact Index Header Record	
Byte	Description
00-03	Pointer to root node
04-07	Pointer to free node list (-1 if not present)
08-11	Reserved for internal use
12-13	Length of key
14	Index options (any of the following numeric values or their sums): <ul style="list-style-type: none"> <li>a. 1 – a unique index</li> <li>b. 8 – index has FOR clause</li> <li>c. 32 – compact index format</li> <li>d. 64 – compound index header</li> </ul>
15	Index signature
16-19	Reserved for internal use
20-23	Reserved for internal use
24-27	Reserved for internal use
28-31	Reserved for internal use
32-35	Reserved for internal use
36-501	Reserved for internal use
502-503	Ascending or descending: <ul style="list-style-type: none"> <li>a. 0 = ascending</li> <li>b. 1 = descending</li> </ul>
504-505	Reserved for internal use
506-507	FOR expression pool length <sup>1</sup>
508-509	Reserved for internal use
510-511	Key expression pool length <sup>1</sup>
512-1023	Key expression pool (uncompiled)

<sup>1</sup>This information tracks the space used in the key expression pool.

Compact Index Interior Node Record	
Byte	Description
00-01	Node attributes (any of the following numeric values or their sums): a. 0 – index node b. 1 – root node c. 2 – leaf node
02-03	Number of keys present (0, 1 or many)
04-07	Pointer to node directly to left of current node (on same level; -1 if not present)
08-11	Pointer to node directly to right of current node (on same level; -1 if not present)
12-511	Up to 500 characters containing the key value for the length of the key with a four-byte hexadecimal number (stored in normal left-to-right format): This node always contains the index key, record number and intra-index pointer. <sup>2</sup> The key/four-byte hexadecimal number combinations will occur the number of times indicated in bytes 02-03.

Compact Index Exterior Node Record	
Byte	Description
00-01	Node attributes (any of the following numeric values or their sums): a. 0 – index node b. 1 – root node c. 2 – leaf node
02-03	Number of keys present (0, 1 or many)
04-07	Pointer to node directly to left of current node (on same level; -1 if not present)
08-11	Pointer to node directly to right of current node (on same level; -1 if not present)
12-13	Available free space in node
14-17	Record number mask
18	Duplicate byte count mask
19	Trailing byte count mask
20	Number of bits used for record number
21	Number of bits used for duplicate count
22	Number of bits used for trail count
23	Number of bytes holding record number, duplicate count and trailing count
24-511	Index keys and information <sup>2</sup>

<sup>2</sup>Each entry consists of the record number, duplicate byte count and trailing byte count, all compacted. The key text is placed at the logical end of the node, working backwards, allowing for previous key entries.

## **Compound Index File Structure (.CDX)**

---

All compound indexes are compact indexes.

One file structure exists to track all the tags in the .CDX file. This structure is identical to the compact index structure with one exception — the leaf nodes at the lowest level of this structure point to one of the tags in the compound index.

All tags in the index have their own complete structure that is identical to the compact index structure for an .IDX file.

## FoxPro 2.0 Project File Structure (.PJX)

Fields	Type	Header	Screen Set	Screen	Program	Menu
NAME <sup>1</sup>	C	text	set name	name	name	name
TYPE	C	H	S	S	P	M
TIMESTAMP	N	time stamp	time stamp	time stamp	time stamp	time stamp
OUTFILE	M	location <sup>2</sup>	output file <sup>1</sup>			output file <sup>1</sup>
HOMEDIR	M	homedir	homedir			homedir
SETID	N	highest id	key number	key number		
EXCLUDE	L		exclude?		exclude?	exclude?
MAINPROG	L		main?		main?	main?
ARRANGED	L			arranged?		
SAVECODE	L	saved?				
DEFNAME	L		defaulted?			
OPENFILES	L		open files?			
CLOSEFILE	L		close files?			
DEFWINDS	L		define winds?			
RELWINDS	L		release winds?			
READCYCLE	L		cycle?			
MULTREAD	L		multiple?			
NOLOCK			lock?			
MODAL	L		modal?			
ASSOCWINDS	M		window list			
DEBUG	L	debug?				
ENCRYPT	L	encrypt?				
NOLOGO			show logo?			
SCRNORDER	N			order number		
SCRNROW	N			vpos		
SCRNCOL	N			hpos		
CMNTSTYLE	N	box/asterisk				
OBJREV	N		obj rev		obj rev	obj rev
COMMANDS	M		bitmap		bitmap	bitmap
DEVINFO	M	developer info				
SYMBOLS	M		symbol table		symbol table	symbol table
OBJECT	M		obj code		obj code	obj code
CKVAL	N					

<sup>1</sup>Includes normalized path.

[illegible]<sup>2</sup>Location of the generated code (<Source>, <Project> or path)

## FoxPro 2.0 Screen File Structure (.SCX)

Fields	Object Types and Field Data							
OBJTYPE	N	SCREEN (1)	WORKAREA (2)	INDEX (3)	RELATION (4)	TEXT (5)	BOX (7)	GROUP (10)
OBJCODE	N	version (10)	1-25	1-25	1-25	SAY (0)	BOX/BOX... (3-6)	
NAME	M	window	file.dbf	file.idx				
EXPR	M		set skip	indexexpr	relexpr	thetext		
VPOS	N	vpos				vpos	vpos	objnumber
HPOS	N	hpos				hpos	hpos	objcount
HEIGHT	N	height				height	height	
WIDTH	N	width				width	width	
STYLE	N	USR/DLG..						
PICTURE	M							
ORDER	M		index name					
UNIQUE	L		current=.T.	unique?				
COMMENT	M							
ENVIRON	L	environ?						
BOXCHAR	C						boxchar	
FILLCHAR	C						fillchar	
TAG	M	title	alias	for expr	into alias			
TAG2	M	footer	tag name	to alias	from alias			
SCHEME	N	schemeno				schemeno	schemeno	
SCHEME2	N	schemeno						
COLORPAIR	N					col pair	col pair	
LOTYPE	N							
RANGELO	M							
HITYPE	N							
RANGEHI	M							
WHENTYPE	N	expr/code						
WHEN	M	when						
VALIDTYPE	N	expr/code						
VALID	M	valid						
ERRORTYPE	N							
ERROR	M							
MESSTYPE	N							
MESSAGE	M							
SHOWTYPE	N	expr/code						
SHOW	M	show						
ACTIVTYPE	N	expr/code						
ACTIVATE	M	activate						
DEACTTYPE	N	expr/code						
DEACTIVATE	M	deactivate						
PROCTYPE	N	expr/code						
PROCCODE	M	proc code						
SETUPTYPE	N	expr/code						
SETUPCODE	M	setup code						
FLOAT	L	float?						
CLOSE	L	close?						
MINIMIZE	L	minimize?						
BORDER	N	border						
SHADOW	L	shadow?						
CENTER	L	center?						
REFRESH	L					refresh?		
DISABLED	L							
SCROLLBAR	L							
ADDALIAS	L	add alias?						
TAB	L							
INITIALVAL	M							
INITIALNUM	N	init obj						
SPACING	N							





## FoxPro 2.0 Report File Structure (.FRX)

Fields	Object Types and Field Data				
OBJTYPE	N	REPORT (1)	BANDINFO (9)	REPFIELD (8)	TEXT (5)
OBJCODE	N	version (0)	TYPE (0-8)	SAY (0)	SAY (0)
NAME	M	window			varname
EXPR	M		group expr	the text	say expr
VPOS	N			vpos	vpos
HPOS	N			hpos	hpos
HEIGHT	N	page len	height	height	height
WIDTH	N	page width		width	width
STYLE	M		style	style	
PICTURE	M			func/pict	
ORDER	M				
UNIQUE	L				
COMMENT	M				
ENVIRON	L	environ?			
BOXCHAR	C				
FILLCHAR	C				
TAG	M				
TAG2	M				
FLOAT	L			float?	float?
STRETCH	L			stretch?	
NOREPEAT	L			norepeat	
RESETRPT	N			reset repeat	
PAGEBREAK	L		pagebreak?		
RESETPAGE	L		reset page?		
SWAPHEADER	L		swap hdr?		
SWAPFOOTER	L		swap ftr?		
EJECTBEFOR	L	eject bef?			
EJECTAFTER	L	eject aft?			
PLAIN	L	plain?			
SUMMARY	L	summary?			
ADDALIAS	L	add alias?			
OFFSET	N	prt offset			
TOPMARGIN	N	top marg			
BOTMARGIN	N	bottom marg			
TOTALTYPE	N			total code	
RESETTOTAL	N			reset code	

[illegible]

## FoxPro 2.0 Label File Structure (.LBX)

Fields	Type	Label Layout	Line Contents	Database Info	Index Info	Relations Info	PDSETUP
OBJTYPE	N	LABEL(30)	(19)	WORKAREA(2)	INDEX(3)	RELATION (4)	
OBJCODE	N			1-25	1-25	1-25	
NAME	M	remarks		data file name	index name		PDSETUP
EXPR	M		label contents		index expr	relational expr	
STYLE	M		print style				
HEIGHT	N	label height					
WIDTH	N	label width					
LMARGIN	N	left margin					
NUMACROSS	N	number across					
SPACESBET	N	spaces between					
LINESBET	N	lines between					
ENVIRON	L	environment saved?					
ORDER	M			index name			
UNIQUE	L				unique?		
TAG	M			alias	for expr	into alias	
TAG2	M			tag name	to alias	from alias	
ADDALIAS	L	add alias?					

## **FoxPro 1.x Report File Structure (.FRX)**

**NOTE:** The .FRX file structure described here is for FoxPro 1.x .FRX files. The structure of FoxPro 2.0 .FRX files is documented earlier in this appendix.

All records in a report file consist of a fixed number of fields that are delimited by tabs (0x09). There is one record for each object type in the report. Each record terminates with a carriage return (0x0D) and a line feed character (0x0A). All information in the file is represented with ASCII characters.

Field #	Object Type with Field Values				
	Screen (1)	Text (5)	Box (7)	Report Field (17)	Band Info (18)
0 <sup>1</sup>	Screen	Text	Box	Report Field	Band Info.
1 <sup>2</sup>	Version <sup>3</sup>	SAY	Box/BoxD/ BoxC	SAY	Band Type <sup>4</sup>
2	Resource File Name	Actual Text	Null	Field Expr.	Group Expr.
3	Heading	Null	Null	Field Type <sup>5</sup>	Null
4	Vertical Position	Vertical Position	Vertical Position	Vertical Position	Page Break <sup>6</sup>
5	Horiz. Position	Horiz. Position	Horiz. Position	Horiz. Position	Reserved
6	Height	Height	Height	Height	Height
7	Width	Width (of actual text)	Width	Width	
8	Font Name	Null	Null	Null	Null
9	Printer Offset		0x00 or ? <sup>7</sup>		Swap Header <sup>6</sup>
10					Swap Footer <sup>5,6</sup>

Field #	Object Type with Field Values				
	Screen (1)	Text (5)	Box (7)	Report Field (17)	Band Info (18)
11					Null
12					Null
13					Null
14					Null
15					Null
16					Null
17	Null	Null	Null		Null
18	Null	Null		Once (0x30)/No duplicate (0x31)	Null
19 <sup>8</sup>	Null	Top/ Float	Top/ Float	Top/Stretch/ Float/Float and Stretch	Null
20	Null	Null	Null	Type Total <sup>9</sup>	Null
21	View File Info	Null	Null	Reset Total <sup>10</sup>	Null
22	Null	Null	Null	Null	Null
23	User Comment	User Comment	User Comment	User Comment	User Comment
24	User Data	User Data	User Data	User Data	User Data

NOTE: Shaded cells represent fields that are reserved for future use. When a field position is reserved, tab delimiters for that position are present but there are no characters in its placeholder. If a field is null, only a position holder is present. There are no characters between the tab delimiters.

**Notes to Object Type with Field Values Table**

<sup>1</sup>Object types and their values in the file:

- |                  |                                 |
|------------------|---------------------------------|
| a. Screen = 0x31 | d. Band Info = 0x31 and 0x38    |
| b. Text = 0x35   | e. Report Field = 0x31 and 0x37 |
| c. Box = 0x37    |                                 |

<sup>2</sup>SAY/GET values:

- |   |
|---|
| a. Single-line box = 0x30 (Box)                                       |
| b. Double-line box = 0x31 (BoxD)                                      |
| c. GET (Input) = 0x32 (GET or BoxC)                                   |
| If the object type is Box, then 0x32 represents a character line box. |
| d. SAY (Output) = 0x34 (SAY)  |

<sup>3</sup>The current version for character-based reports is "1100".

<sup>4</sup>Band types and values when the object type is Band Info:

- |                        |                        |
|------------------------|------------------------|
| a. Title = 0x30        | e. Group Footer = 0x35 |
| b. Page Header = 0x31  | f. Page Footer = 0x36  |
| c. Group Header = 0x33 | g. Summary = 0x37      |
| d. Detail = 0x34       |                        |

<sup>5</sup>Field types:

- |               |             |
|---------------|-------------|
| N - Numeric   | D - Date    |
| M - Memo      | L - Logical |
| C - Character |             |

<sup>6</sup>When the object type is Band Info and the Band Type is Group Header, 0 = off and 1 = on. In all other cases, this field is null.

<sup>7</sup>Font size for a Box object type is 0x30 if it is either a single- or double-line box. If it is a character box, the binary value of the character used is shifted eight bits to the left and its value is placed in the field as base 10-ASCII characters.

<sup>8</sup>Object type and flag values:

- |                   |                             |
|-------------------|-----------------------------|
| a. Top = 0x30     | c. Float = 0x33             |
| b. Stretch = 0x31 | d. Float and Stretch = 0x34 |

<sup>9</sup>Values for Totaling option (ASCII character of number in file):

- |                  |                |
|------------------|----------------|
| a. No total = 0  | d. Average = 3 |
| b. Count = 1     | e. Minimum = 4 |
| c. Sum total = 2 | f. Maximum = 5 |

<sup>10</sup>Values for Reset Total option (ASCII character of number in file):

- |                      |                        |
|----------------------|------------------------|
| a. End of report = 0 | c. End of column = 2   |
| b. End of page = 1   | d. End of group = 3-22 |

## **FoxPro 1.x Label File Structure (.LBX )**

---

**NOTE:** The .LBX file structure described here is for FoxPro 1.x .LBX files. The structure of FoxPro 2.5 .LBX files is the same as that of .FRX files documented earlier in this appendix.

Label files contain information for one label definition file as defined by the user. All bytes are stored in Intel 8086 format.

<b>Byte</b>	<b>Description</b>
00	Version: 03 = FoxPro label
01-60	Remarks (ASCII characters)
61-62	Height: Number of lines in label
63-64	Left Margin: Column number of left margin
65-66	Width: Width of label
67-68	Number Across: Number of labels across row
69-70	Spaces Between: Number of spaces between labels
71-72	Lines Between: Number of lines between labels
73-74	Length of expression contents
75-n	Label Expression Contents: List of expressions in label separated by carriage returns (0x0D)



## **FoxBASE+ Memo File Structure (.DBT)**

---

FoxBASE+ memo files don't have the versatility of FoxPro memo files. They can contain only ASCII text data.

Records output to this file in blocks are 512 bytes in size. The block at file position zero contains the block number of the first free file position. This block number is stored in the first two bytes in reverse order (Intel 8086 format). To find the address of the next free block, multiply the size of a single block (512 bytes) by the block number.

The blocks that follow the initial block information contain the text of the memos from the associated table. The memo field in the table file contains the number of the block in the memo file that contains the actual text. All memo blocks start on 512 byte boundary addresses.

**FoxPro Macro File Format (.FKY)**

---

File Header	
Byte	Description
01-03	Signature, Hex 79ff
04-15	Ignored
16-17	Number of macros (binary)
18-end	The macros
Individual Macros	
00-19	Macro name
20-21	Macro length (in keystrokes, binary)
22-23	Keystroke (two bytes, binary)
24-end	Macro keystrokes