KMSYS
WORLDWIDE, INC

WinQ User Guide

For T27 and UTS eXpress Enterprise SDK

10 August 2011

The information contained in this document is the latest available at the time of preparation; therefore, it may be changed without notice, and it does not represent a commitment on the part of KMSYS Worldwide, Inc. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy this software on magnetic tape, disk, or any other medium for any purpose other than stated in the terms of the agreement, or without the express written permission of KMSYS Worldwide, Inc.

This material constitutes proprietary and confidential property of KMSYS Worldwide, Inc., having substantial monetary value and is solely the property of KMSYS Worldwide, Inc. This property is disclosed to the recipient thereof in confidence only and pursuant to the terms and conditions and for the purpose set forth in written agreements by and between KMSYS Worldwide, Inc., and the recipient of this material.

If you have any comments about the software or documentation, notify KMSYS Worldwide, Inc., in writing at the following address:

KMSYS Worldwide, Inc.
P.O. Box 669695
Marietta, Georgia 30066
U.S.A.

Technical Support (770) 635-6363 - Main Number (770) 635-6350 - Fax (770) 635-6351

WinQ Release 1R1, April 2000

eQuate, Host Gateway Server, I-QU PLUS-1, I-QU ReorgComposer, InfoQuest, InfoQuest Client, Q-LINK, QPlex, QPlexView, T27 eXpress Enterprise, UTS eXpress Enterprise, T27 eXpress IT, T27 eXpress Net, T27 eXpress Plus, T27 eXpress Pro, UTS eXpress IT, UTS eXpress Net, UTS eXpress Plus, UTS eXpress Pro and WinQ are trademarks or registered trademarks of KMSYS Worldwide, Inc. Microsoft, Windows, Visual Studio, Visual Basic and Visual C++ are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Delphi is a trademark of Borland International. Sperry, Unisys, UTS, UNISCOPE and MAPPER are trademarks of Unisys Corporation. Enable is a trademark of Cypress Software, Inc. All other trademarks and registered trademarks are the property of their respective owners.

RESTRICTED RIGHTS LEGEND

If this Product is acquired by or for the U.S. Government then it is provided with Restricted Rights. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, or subparagraphs (c)(1) and (2) of the Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, or clause 18-52.227-86(d) of the NASA Supplement to the FAR, as applicable.

Table of Contents

Chapter 1: WinQ Introduction and Installation

WinQ is distributed with T27 eXpress Enterprise and UTS eXpress Enterprise. WinQ is a powerful development tool, consisting of ActiveX Controls (OCXs) that allows Windows applications to access Unisys 2200 and MCP Series Applications.

1.1 System Requirements

Any currently supported level of Microsoft Windows.

Any PC hardware platform capable of running Microsoft Windows, with at least 4MB of RAM (a minimum of 8MB is recommended).

Approximately 600KB of available disk space to contain the OCX.

1.2 Supported Software

WinQ can be use in any Windows application that supports ActiveX Controls. Examples are Visual Studio, Visual Basic, C++, Delphi and Power Builder.

Windows applications can use the WinQ ActiveX Controls to access data presented by legacy MCP and OS2200 host applications without requiring any changes to the host programs.

WinQ also provides an optional relocatable library for the 2200 that supports calls from ASCII COBOL programs. This library is only required if you plan to use the WinQ Application Operation Mode.

1.3 Who Should Use This Manual?

This manual is intended for use by application developers who are familiar with Microsoft Windows application programming with Visual Basic and ASCII COBOL.

1.4 What Has Changed Since Last Publication?

Changes since last publication are noted with a vertical bar to the right of the entry as shown on this paragraph.

1.5 What Does This Manual Contain?

The chapters in this manual describe WinQ from general overview through details of each WinQ API function. A brief description of the contents of each chapter follows:

Chapter 1: WinQ Introduction and Installation

This chapter is brief introduction to WinQ describing supported software, defining contents and usage of this manual, and specifying the steps required to install WinQ.

Chapter 2: WinQ Structure

This chapter presents the WinQ structure and includes descriptions of the three WinQ operation modes and sign-on script capability.

Chapter 3: Developing a WinQ Application

This chapter describes the step for developing a WinQ Application.

Chapter 4: WinQ Component Reference

This chapter shows the properties, methods and events available thorough the WinQ OCX libraries.

Chapter 5: Screen Operation Mode Example

This chapter presents an example of a Visual Basic application developed to access a legacy host application using the WinQ Screen Operation Mode.

Chapter 6: Application Operation Mode Procedures

This chapter outlines the 2200 procedures that are used when writing host programs to control WinQ operations.

Chapter 7: Application Considerations

This chapter presents a discussion of special considerations that must be dealt with when developing WinQ client-server applications.

Chapter 8: WinQ Host Subroutine Library

This chapter provides a detailed explanation of the WinQ interface to the host application program including the calls to the WinQ 2200 subroutines.

Chapter 9: Application Operation Mode Example

This chapter provides an example of a complete WinQ application using the WinQ Application Operation Mode including. Both the client and server programs are shown.

Chapter 10: CUSTSCHEMA Schema Listing

This chapter contains the schema listing used in the example.

1.6 Installation

The WinQ OCX components are installed when either UTS eXpress Enterprise or T27 eXpress Enterprise SDK are installed; therefore, no additional installation steps are required on the PC side.

If your application will use the WinQ Application Operation Mode (see Chapter 3) to interface with a 2200 application, you will need to install the WinQ Relocatable Library on the 2200 host.

The WinQ Relocatable Library may simply be copied to the 2200 host using the @COPY,G ECL command. Optionally, the file may be installed and registered with the collector by performing a "LOCAL" install with COMUS. The following example illustrates both:

```
@ASG,TJ        T.,U9S,WQREL
@ASG,T         TPF$.,///256
@COPY,G        T.,
@QUAL          your-COMUS-database-qualifier
@ADD           WINQ/INSTALL
```

The WINQ/INSTALL element contains the following:

```
@ASG,UPV       SYS$LIB$*WINQ(+1).,F///256
@COPY          ,SYS$LIB$*WINQ(+1).
@FREE          SYS$LIB$*WINQ(+1).
@CHG,V         SYS$LIB$*WINQ.
@COMUS
INSTALL LOCAL,SYS$LIB$*WINQ.COMUS$SGS
```

The COMUS$SGS element contains the following SGSs:

```
PRODUCT WINQ,1R1 RELFILE
FILE SYS$LIB$*WINQ,256 NOTWRITEABLE NOROLLOUT
COLLECTOR SYS$LIB$*WINQ 6
```

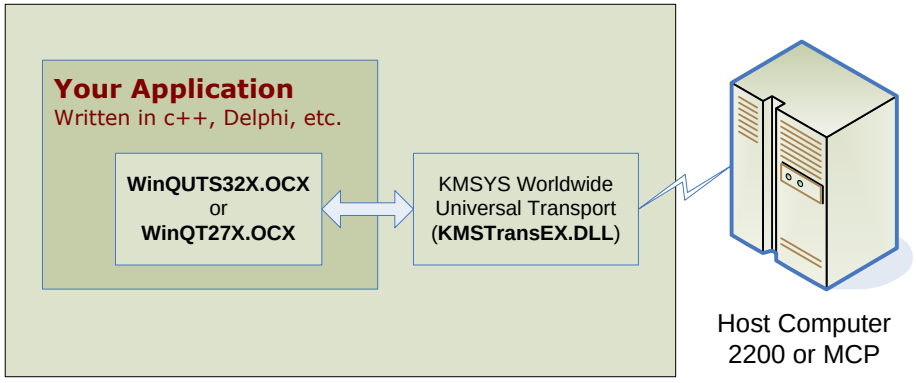
Chapter 2: WinQ Structure

The WinQ interface consists of two ActiveX controls:

WinQUTS32x.OCX, distributed with **UTS eXpress Enterprise SDK**, for accessing Unisys 2200 host computers.

WinQT27x.OCX, distributed with **T27 eXpress Enterprise SDK**, for accessing Unisys MCP host computers.

Both controls can be used either separately or simultaneously in a single application.



2.1 WinQ Operation Modes

WinQ supports three operation modes: Screen, Raw Message and WinQ Application. The Raw Message and WinQ Application modes are only available in WinQUTS32X for OS 2200 host environments.

2.1.1 Screen Operation Mode

Screen Operation mode is the most basic way of using WinQ and the best way of using WinQ to interface with legacy host application. In this mode, there is usually no need to change the host application to interface with a WinQ PC application. In screen mode, your application uses an internal terminal screen in much the same way a user uses a visible terminal screen. It is like having a terminal emulator running inside your application. WinQ manages the terminal session while your application has access to the screen and terminal

keyboard functions. The term "screen scraper" is often used to describe this mode of operation.

2.1.2 Raw Message Operation Mode

Raw Message Operation mode bypasses the internal terminal emulation and uses messages received from and sent to the 2200 host. The host application needs to be aware of the message format as determined by you, the developer.

Note: This mode is only available for Unisys 2200 systems.

2.1.3 WinQ Application Operation Mode

WinQ Application Operation mode is the most advanced and requires the WinQ Application interface to be installed and used by the 2200 host application program. WinQ Application Operation mode provides extra benefits like built-in file transfer functions.

Note: This mode is only available for Unisys 2200 systems.

In this mode, WinQ handles ALL communications between the host server application and the PC/Workstation client application. WinQ can send and receive single-block text messages and automatically send and receive text or binary files of any length.

2.2 Scripting in WinQ

The WinQ component supports the same scripting language used with UTS and T27 eXpress Enterprise. This feature allows the application to perform variable external functions. Such a function would be establishing the host session, processing the user's sign-on and start the host application. The sign-on function is often different from user to user. By using scripts, it is possible to develop and use different sign-on scripts without having to develop complex code within the WinQ application program.

Note: See the eXpress Script Editor PDF (**ExpScriptEditor.pdf**) or the on-line help for the eXpress Script Editor for a full list of functions and subroutines.

Chapter 3: Developing a WinQ Application

This chapter describes in more detail the development steps required when utilizing the three operation methods.

3.1 WinQ Screen Operation Mode

As described earlier WinQ Screen Operation Mode (SOM) is the most basic way of using WinQ to interface a Windows program with a legacy host application. In Screen Mode, your Windows program uses an internal terminal emulator provided by WinQ, in much the same way a human user uses a terminal or terminal emulator. Because your application is using a terminal interface there is usually no need to modify host application programs or screens.

The WinQ Component provides several properties and methods that allow your program to interact with the internal terminal emulator. If desired, your application can even show the terminal emulation window, allowing both your program and the user to access it simultaneously (also handy as a debugging tool).

3.1.1 Using Screen Operation Mode

To use WinQ SOM, your application needs to perform the following basic steps:

1. Establish a host connection
2. Process the user's sign on (optional)
3. Interactively run the host application using the internal terminal emulator
4. Close the host connection

3.1.1.1 Establish a Host Connection

Establishing the host connection requires configuring an eXpress Enterprise screen name on the local computer using the UTS or T27 eXpress Enterprise Component Configuration program. Note: The screen name may be one that also is configured by the T27 or UTS eXpress Enterprise emulator.

Once a screen name is configured, use the WinQ component's **OpenSession** method to connect to the host.

3.1.1.2 Process User's Sign-On (Optional)

This process is optional, because sometimes just establishing a host session makes the host ready to begin executing the application. This is often the case in some older installations. In most cases, there will be a need to validate your user's access to the host application.

The user sign-on is best handled by a script executed by the WinQ component's **RunScript** method. Scripts can be developed to meet the need of any user or application.

3.1.1.3 Interactively Run the Host Application

Once the session is in a state to run the host application (connection established and user signed on), your application does what the user would do using a terminal directly. WinQ provides several properties, methods and events to allow your application program to see and use the internal terminal emulator.

3.1.1.4 Close the Host Connection

Closing the session must be done before closing your application. Use the WinQ component's **CloseSession** method to close the host connection.

3.2 WinQ Raw Message Operation Mode

In this mode, the same process are required to connect to the host (2200 only) sign on, run the host application and close the host connection. The difference is that the communications between the Windows application and the host application is all completed by sending and receiving raw messages. The WinQ methods used are the **SendRaw** and **GetMessage** functions (see the following chapter).

3.3 WinQ Application Operation Mode

WinQ Application operation mode is only available to OS/2200 users and requires the WinQ library be installed on the host.

In this mode, the same processes are required to connect to the host (2200 only), sign on, run the host application and close the host connection. The difference is that the communications between the Windows application and the host application is all completed by sending and receiving formatted WinQ application messages using the **GetApp** and **SendApp** functions in the Windows program and the **WINQ_SVRGET** and **WINQ_SVRPUT** subroutine calls in the host program.

Chapter 4: WinQ Component Reference

The WinQ custom controls (WinQUTS32X and WinQT27X) are provided to allow the Windows application program to interface with the internal terminal emulator, and in some cases, directly with the host program. Procedures are provided to open and close the session, perform data and terminal keystrokes, get screen attributes and colors, obtain screen and line text, run scripts, set screen cursor position and text, etc.

When operating in Raw Message Mode or Application Mode (WinQUTS32X only), methods are provided that allows the Windows application to communicate with a host program through the terminal message queue rather than the terminal screen.

4.1 Custom Controls

Like most controls, the WinQ controls have properties, events and methods that apply to WinQ. To include a WinQ custom control in your application, you must add the file WinQUTS32X.OCX or WinQT27X.OCX to your project (e.g., in Visual Basic, use the **Custom Controls** selection from the **Tools** menu). These files will have been installed and registered with Windows when you installed eXpress Enterprise SDK.

4.2 Properties

The following properties of the WinQ control may be examined and set depending upon whether the access is Read or Write, respectively.

Note: WinQ properties are set at runtime, not at design-time.

Name	Type	Access	Description
ActivateTransportTrace	Integer (0/1)	Write	Activate KMS Univeral Transport trace to file. This property must be set before the **OpenSession** method is called. Setting this property at any other time has no effect.
Columns	Integer	Read only	The number of columns in the current internal terminal screen.
CurrentRoute	Wide String	Read only	The Screen name currently opened. Set after **OpenSession** method is called.
CursorColumn	Integer	Read only	The current cursor row in the internal terminal screen; 0 when a session is not open.
CursorRow	Integer	Read only	The current cursor column in the internal terminal screen; 0 when a session is not open.

Name	Type	Access	Description
KeyboardLocked	Integer	Read only	Indicates when the internal terminal's keyboard has been locked. (WinQUTS32X only)
LastStatus	Integer	Read only	Last WinQ operation status code (see Section 4.6.3, WinQ Status Codes). Check this values if a method returns an error result.
LanguageType	Integer (0/1)	Read/ Write	0 = Latin, English, Western Europe, etc. 1 = Mainland Chinese, simplified. Default = 0. (WinQUTS32X only)
MessageWait	Integer	Read only	Returns 1 if message waiting is on; 0, if off. (WinQUTS32X only)
OperationMode	Integer (0/1/2)	Read/ Write	Indicates the current WinQ operation mode: 0 Screen (default), 1 Raw Messages (WinQUTS32X only), 2 WinQ Application (WinQUTS32X only).
Page	Integer	Read/ Write	Sets or gets the current page within the screen environment. When setting the page, the value must be within one (1) through the number of pages configured. Note: The **Page** property is in WinQUTS32X only for reasons of compatibility with WinQT27X and, thus, is always set to 1.
Pages	Integer	Read only	Gets the number of pages configured in the screen environment. The **Pages** property is in WinQUTS32X only for reasons of compatibility with WinQT27X and, thus, is always set to 1.
PrintFontBold	Integer	Read/ Write	Set the print font to the bold style(1= true, 0 = false). This property only affect screen printing executed by the WinQ component either by a print command received from the host or a print screen key issued by the **DoTerminalKey** method. (WinQUTS32X only)
PrintFontItalic	Integer	Read/ Write	Set the print font to the italic style(1= true, 0 = false). This property only affect screen printing executed by the WinQ component either by a print command received from the host or a print screen key issued by the **DoTerminalKey** method. (WinQUTS32X only)
PrintFontName	String	Read/ Write	Set the print font name. This property only affect screen printing executed by the WinQ component either by a print command received from the host or a print screen key issued by the **DoTerminalKey** method. (WinQUTS32X only)

Name	Type	Access	Description
PrintFontSize	Integer	Read/ Write	Set the print font size. This property only affect screen printing executed by the WinQ component either by a print command received from the host or a print screen key issued by the **DoTerminalKey** method. (WinQUTS32X only)
PrintOrientation	Integer	Read/ Write	Set the print orientation(1= Portrait, 2 = Landscape). This property only affect screen printing executed by the WinQ component either by a print command received from the host or a print screen key issued by the **DoTerminalKey** method. (WinQUTS32X only)
Rows	Integer	Read only	The number of rows in the current internal terminal screen.
ScreenFontBold	Integer	Read/ Write	Set the screen font to the bold style(1= true, 0 = false). This property just affects the WinQ terminal screen when visable. (WinQUTS32X only)
ScreenFontItalic	Integer	Read/ Write	Set the screen font to the italic style(1= true, 0 = false). This property just affects the WinQ terminal screen when visable. (WinQUTS32X only)
ScreenFontName	String	Read/ Write	Set the screen font name. This property just affects the WinQ terminal screen when visable. (WinQUTS32X only)
ScreenFontSize	Integer	Read/ Write	Set the screen font size. This property just affects the WinQ terminal screen when visable. (WinQUTS32X only)
ScreenVisible	Integer (0/1)	Read/ Write	Indicates weather or not the internal screen window is visible. Set this property to 1 to make the screen window visible and accessible to the user. Set it to 0 (default) to make the screen invisible and inaccessible to the user. Tip: Set the ScreenVisible property to one when debugging. When the internal screen is visible, the screen has a mouse pointer row and column indicator (see "Prt" on the status bar at the bottom of the window) that is useful in WinQ Screen Operation Mode when determining the row and colums of specific fields on the screen. This indicator acts differently from the cursor pointer in that it is not dependent upon whether or not a part of the screen is protected from user access.
SessionOpen	Integer (0/1)	Read only	Indicates that a host session is currently open.

Name	Type	Access	Description
TraceOption	Integer (0/1/2/3)	Read only	Sets the type of message trace desired: 0 = none 1 = trace to window 2 = trace to file 3 = trace to both window and file. Note: The **TraceOption** should only be set after the successful completion of the **OpenSession** method. Tip: Use the "trace to window" option as a debugging tool where you may view all comunications between the host program and the PC application.
WantRawMessages	Integer (0/1)	Read/ Write	Indicates that your application wants to process **OnRawMessageReceived** events.
WantScreenUpdated	Integer (0/1)	Read/ Write	Indicates that your application wants to process **OnScreenUpdated** events.
WantTerminalKeyInput	Integer (0/1)	Read/ Write	Indicates that the application wants to receive and process **OnTerminalKeyInput** events.

4.3 Methods

The following methods are used to control the connection to the host, interface with the internal emulator screen, run scripts and communicate with host programs.

4.3.1 CloseSession

Format:

Function CloseSession() as Integer

Description:

Close the current host session.

Returns:

If successful, the function returns a 1; else, it returns a 0.

Example:

```
WinQ32.CloseSession
```

4.3.2 DoDataKey

Format:

Sub DoDataKey (CharCode as Integer)

Description:

Enter a data key into the internal terminal emulator as it if where typed from the keyboard by an end user. **CharCode** is the ASCII character value (e.g., A = 65).

Example:

```
 WinQ32.DoDataKey 65   ' Enter an "A"
```

4.3.3 DoTerminalKey

Format:

Sub DoTerminalKey (KeyCode as Integer)

Description:

Cause the specified Terminal Key sequence to be executed by the internal terminal emulator. All terminal key functions are available. See Section 4.6, *KeyCode Constant*s for the terminal type currently being emulated.

Note: Use the **DoTerminalKey** method to transmit to the host, send a function key, erase display, etc.

Example:

```
WinQ32.DoTerminalKey UK_ERASE_TO_END_DISPLAY
```

4.3.4 GetApp

This method is only available in WinQUTS32X with **OperationMode** 2 (app).

Format:

Function GetApp (TimeOut as Integer) as String

Description:

Get the next WinQ Application message from the message queue. If there are no messages currently in the message queue, WinQ will wait for a new message until the **TimeOut** expires.

If an error occurs or the **TimeOut** expires, an empty string will be returned and **LastStatus** will contain the WinQ error status. If a message is successfully retrieved, the message string will be returned.

Returns:

If successful, the function returns a string containing the application message; else, an empty string is returned.

Example

```
Reply = CustMain.WinQ32.GetApp(Time_Out)
```

4.3.5 GetFile

This method is only available in WinQUTS32X with **OperationMode** 2 (app).

Format:

Function GetFile (HostFile as String, PCFile as String, FileType as Integer, TimeOut as Integer) as Integer

Description:

Retrieve a file from the host using the WinQ Application interface. The WinQ Application session must be established.

Returns:

If the file transfer is successful, this function returns a 1; else, it returns a 0 and **LastStatus** will contain an error code.

Example:

```
Rslt = WinQ32.GetFile("WORKFILE", PC_File, 1, 300)
```

4.3.6 GetMessage

This method is only available in WinQUTS32X with **OperationMode** 1 or 2 (raw or app).

Format:

Function GetMessage (TimeOut as Integer) as String

Description:

Retrieve the next raw message from the host message queue.

Returns:

If a message is in the queue or is received before the **TimeOut** expires, the message string is returned and **LastStatus** is set to 0. If no message is received before the **TimeOut** or an error is encountered, **LastStatus** will contain an error status code and an empty string will be returned.

4.3.7 GetScreenAttributes

Format:

Function GetScreenAttributes (Row as Integer, Column as Integer) as Integer

Description:

Retrieve certain screen attributes at the specified screen row and column coordinates. The following attribute values may be returned:

Constant	Value	Description
SATTR_NORMAL	0	Normal field
SATTR_FIELD	1	Start of field (set on 1st position of field)
SATTR_TAB	2	Tab stop (at start of field only)
SATTR_PROTECTED	8	Protected-Output only
SATTR_VIDEO_OFF	16	Video off (Data is present in screen buffer
SATTR_BLINK	128	Blinking field
SATTR_RIGHT	256	Right justified data
SATTR_REV	1024	Reverse video
Unique T27 Attributes:		
SATTR_ULINE	4	T27 Underline
SATTR_BRIGHT	512	T27 Bright
Unique UTS Attributes		
SATTR_CHANGED	4	Data field changed flag
SATTR_NUMERIC	32	Numeric only input
SATTR_ALPHA	64	Alpha only input
SATTR_LOWINT	512	UTS Low intensitity

Returns:

The function returns the attribute bits. If an error is encountered, a 0 is returned.

4.3.8 GetScreenColors

Format:

Function GetScreenColors (Row as Integer, Column as Integer) as Integer

Description:

Retrieve the color attributes of the screen at the specified screen row and column.

Colors are expressed as an index in the range 0 to 7. The first digit is the background color index and the second is foreground color index.

Colors: 0 = black, 1 = Red, 2 = Green, 3 = Yellow, 4 = Blue, 5 = Magenta, 6 = Cyan, 7 = white.

For example, white text on a red background is 17 hexadecimal or 23 decimal.

Returns:

This function returns the color attribute code. If an error was encountered, a 0 is returned.

4.3.9 GetScreenLine

Format:

Function GetScreenLine (Row as Integer) as String

Description:

Retrieve an entire line of text, including leading and trailing spaces, from the internal screen at the specified row.

Returns:

If successful, the function returns the line as a string; otherwise, it returns an empty string and status will contain an error code.

4.3.10 GetScreenText

Format:

Function GetScreenText (Row as Integer, Column as Integer, Length as Integer) as String

Description:

Retrieve a string of text, including leading and trailing spaces, from the internal screen at the specified row and column and for the specified length.

Returns:

If successful, the function returns the text as a string; otherwise, it returns an empty string and status will contain an error code.

Example:

```
If WinQ32.GetScreenText(1, 1, 5) = "CUST6" Then ' Orders Screen???
```

4.3.11 InterruptOperation

Format:

Function InterruptOperation (timeOut as Integer) as Integer

Description:

Interrupt the current WinQ operation. This method only applies only to the **GetMessage** and **GetApp** functions.

Returns:

Returns 1 if successful, else returns 0.

4.3.12 OpenSession

Format:

Function OpenSession (Route as String) as Integer

Description:

Open a host session using the specified eXpress Connect Route. The route must be configured correctly using the eXpress Connect Visual Configuration.

Returns:

This method returns a 1 if successful; else, it returns a 0.

Example:

```
WinQ32.OpenSession(SessionSettings.Txt_Route.Text)
```

4.3.13 RefreshScreen

Format:

Sub RefreshScreen

Description:

Cause the WinQ Internal screen to be refreshed immediately. This method is only effective if the terminal screen is visible. It should be called whenever the application modified the content of the screen.

Example:

```
WinQ32.RefreshScreen
```

4.3.14 RunScript

Format:

Function RunScript (ScriptFile as String, PCnt as Integer, P1 as String, P2 as String, P3 as String, P4 as String, P5 as String, P6 as String) as Integer

Description:

Run the specified Enable Script passing the parameter values specified in P1 through P6. The specified script may be either an Enable script source file (*.BAS) or a compiled script file (*.BAX).

Returns:

Returns the result returned by the script (see the **ScriptResult** property on the **TTermScreen** class under **eXpress Scripting Classes** in the eXpress Script Editor Help or PDF).

Example:

```
Rslt = WinQ32.RunScript(SessionSettings.Txt_Script, 2, Trim(SessionSettings.Txt_UserId),
Trim(SessionSettings.Txt_Password), "", "", "", "")
```

4.3.15 ScreenBounds

Format:

Sub ScreenBounds (Left as Integer, Top as Integer, Width as Integer, Height as Integer)

Description:

Sets the internal terminal screens position. Note: If any parameters are negative, they will be ignored.

4.3.16 SendApp

This method is only available in WinQUTS32X with **OperationMode** 2 (app).

Format:

Function SendApp (Msg as String, TimeOut as Integer) as Integer

Description:

Transmit a WinQ Application formatted message to the host.

Returns:

Returns 1 if successful, else returns 0 and **LastStatus** is set to a WinQ error code.

Example:

```
Rslt = CustMain.WinQ32.SendApp(Msg, Time_Out)
```

4.3.17 SendFile

This method is only available in WinQUTS32X with **OperationMode** 2 (app).

Format:

Function SendFile (HostFile as String, PCFile as String, Filetype as Integer, TimeOut as Integer) as Integer

Description:

Transmit a complete file to the host.

Returns:

Returns 1 if successful, else returns 0 and **LastStatus** contains a WinQ error code.

4.3.18 SendRaw

This method is only available in WinQUTS32X with **OperationMode** 1 or 2 (raw or app).

Format:

Function SendRaw (Msg as String, TimeOut as Integer) as Integer

Description:

Transmit a raw message to the host.

Returns:

Returns 1 if successful, else returns 0 and **LastStatus** contains a WinQ error code.

4.3.19 SetCursorPosition

Format:

Sub SetCursorPosition (Row as Integer, Column as Integer)

Description:

Move the screen cursor to the specified row and column in the internal screen.

Example:

```
WinQ32.SetCursorPosition 11, 17
```

4.3.20 SetScreenFont

Format:

Sub SetScreenFont (Name as String, Size as Integer, Bold as Integer)

Description:

Allow the application to change the internal screen's display font. This method is only effective when the internal screen is visible.

4.3.21 SetScreenText

Format:

Sub SetScreenText (Row as Integer, Column as Integer, Text as String)

Description:

Replace text in the internal screen starting at the specified row and column.

Example:

```
WinQ32.SetScreenText 8, 17, "Q"
```

4.4 Events

The following events occur upon receiving communications from the host:

4.4.1 OnPrinterDeselected

Format:

Sub OnPrinterDeselected

Description:

The **OnPrinterDeselcted** event is fired when the printer is deselected (see **OnPrinterSelected**, below).

4.4.2 OnPrinterSelected

Format:

Sub OnPrinterSelected

Description:

The **OnPrinterSelected** event is fired when a raw message containing printer device selection is detected. This event, along with the **OnPrinterDeselected** event, allows a WinQ programmer to determine when messages from the host are to go to

a printer. The programmer may capture printer bound messages in the **OnRawMessageReceived** and handle them programmatically.

4.4.3 OnRawMessageReceived

Format:

Sub OnRawMessageReceived (Msg as String, ByRef MapIt as Integer)

Description:

The **OnRawMessageReceived** event is fired when a new message is received from the host, before it is translated and mapped to the internal screen. Msg contains the entire message as received from the host, including all control sequences.

By default, the message will be translated and mapped to the internal screen once your event handler exits; however, if you do not want the message to be translated and mapped to the internal screen, you may set MapIt to 0 (false). If MapIt is not set to 0, the **OnScreenUpdate** event will also be fired.

4.4.4 OnScreenUpdated

Format:

Sub OnScreenUpdated

Description:

The **OnScreenUpdated** event is fired every time the internal screen is updated by a message, or messages, received from the host. Screen updates made by WinQ methods do not cause this event to be called.

To get this event the **WantScreenUpdated** property must be set to 1.

4.4.5 OnTerminalKeyInput

Format:

Sub OnTerminalKeyInput(Key as Integer, ShiftStates as Integer, ByRef Handled as Integer)

Description:

The **OnTerminalKeyInput** event is fired when a key sequence is typed into the terminal screen directly, before it is processed by the internal terminal. Key contains the virtual key code of the key sequence typed. **ShiftStates** contains the state of the Shift, Alt and Ctrl keys.

ShiftStates Values:
Shift Key = 1
Alt Key = 2
Ctrl Key = 4

These codes may be OR'ed together to make combinations. For example Ctrl+Shift = 5, Alt+Ctrl = 6 or Shift+Alt+Ctrl = 7.

The **Handled** parameter allows the program to control whether or not the key is processed by the internal terminal. Set it to zero (default) if the internal terminal is to process the key, or 1 if the key is to be ignored by internal terminal.

Note: The **WantTerminalKeyInput** property must be set to 1 to get this event.

4.5 KeyCode Values

KeyCode values will vary depending on whether you are using WinQUTS32.OCX or WinQT27.OCX. The following files containing constant declarations for both versions are installed with the product files:

WinQCC.H - Declarations for C and C++
WinQDlp.PAS - Declarations for Delphi (or Pascal)
WinQVB.BAS - Declarations for Visual Basic

These files also contain declarations for screen attribute values.

4.6 KeyCode Constants

4.6.1 UTS Keyboard Functions

Constant	Value
UK_BACK_SPACE	95
UK_CURSOR_DOWN	6
UK_CURSOR_LEFT	7
UK_CURSOR_RETURN_KEY	32
UK_CURSOR_RIGHT	8
UK_CURSOR_TO_END_LINE	66
UK_CURSOR_TO_HOME	23
UK_CURSOR_TO_START_LINE	65
UK_CURSOR_UP	9
UK_DELETE_IN_DISPLAY	11
UK_DELETE_IN_LINE	12
UK_DELETE_LINE	10
UK_ERASE_CHAR	67
UK_ERASE_DISPLAY	14
UK_ERASE_TO_END_DISPLAY	15
UK_ERASE_TO_END_FIELD	16
UK_ERASE_TO_END_LINE	17
UK_FKEY_1	43
UK_FKEY_10	52
UK_FKEY_11	53
UK_FKEY_12	54
UK_FKEY_13	55
UK_FKEY_14	56
UK_FKEY_15	57
UK_FKEY_16	58
UK_FKEY_17	59
UK_FKEY_18	60
UK_FKEY_19	61
UK_FKEY_2	44
UK_FKEY_20	62
UK_FKEY_21	63
UK_FKEY_22	64
UK_FKEY_3	45
UK_FKEY_4	46
UK_FKEY_5	47
UK_FKEY_6	48
UK_FKEY_7	49
UK_FKEY_8	50
UK_FKEY_9	51
UK_INSERT_IN_DISPLAY	25

Constant	Value
UK_INSERT_IN_LINE	26
UK_INSERT_LINE	24
UK_KEYBOARD_UNLOCK	27
UK_LINE_DUP	28
UK_MSG_WAIT	29
UK_PRINT_ENTIRE_SCREEN	69
UK_PRINT_KEY	30
UK_SOE	3
UK_TAB_BACK	33
UK_TAB_FORWARD	34
UK_TAB_SET	35
UK_TRANSMIT_KEY	36

4.6.2 T27 Keyboard Functions

Constant	Value
TK_ARROWDN	249
TK_ARROWLEFT	247
TK_ARROWRIGHT	248
TK_ARROWUP	246
TK_BACKSPACE	8
TK_BACKTAB	196
TK_BOUND	218
TK_CARRIAGERTN	13
TK_CLRALLVTAB	16442
TK_CLREOL	134
TK_CLREOP	135
TK_CLRFORMS	159
TK_CLRHOME	128
TK_COPY	16432
TK_CTRL	164
TK_CUT	16431
TK_DBLZERO	234
TK_DELCHAR	132
TK_DELCHARPAGE	16425
TK_DELLINE	133
TK_HOME	174
TK_INSCHAR	130
TK_INSCHARPAGE	16424
TK_INSLINE	131
TK_LOCAL	168
TK_LOCKCTRL	165
TK_LOGICALEOL	16415
TK_MARK	217
TK_MOVELINEDOWN	138

Constant	Value
TK_MOVELINEUP	139
TK_NEXTPAGE	253
TK_PASTE	16434
TK_PREVEPAGE	252
TK_PRINTALL	157
TK_PRINTUNPROT	156
TK_RECALL	214
TK_RECEIVE	170
TK_ROLLDN	136
TK_ROLLUP	137
TK_SETFORMS	158
TK_SPECIFY	166
TK_STORE	213
TK_TAB	198
TK_TOGGLEFORMS	141
TK_TOGGLETAB	16441
TK_TRANSMIT	172
TK_TRANSMITLINE	16428
TK_TRIPZERO	236
TK_UPPERONLYON	210
TK_UPPERONLYOFF	211
TK_WRITEESC	16426
TK_WRITEETX	3
TK_WRITEGS	16427

4.6.3 WinQ Status Codes

Code	Description
0	OK
1	TimeOut - operation timed out.
2	WrongMode - wrong operation mode for this method.
3	FileNotFound - upload file does not exist.
4	FileOpenError - download file can't be created.
5	HostError - the host responded with an error.
6	NoSession - no session is opened.

Chapter 5: Screen Operation Mode Example

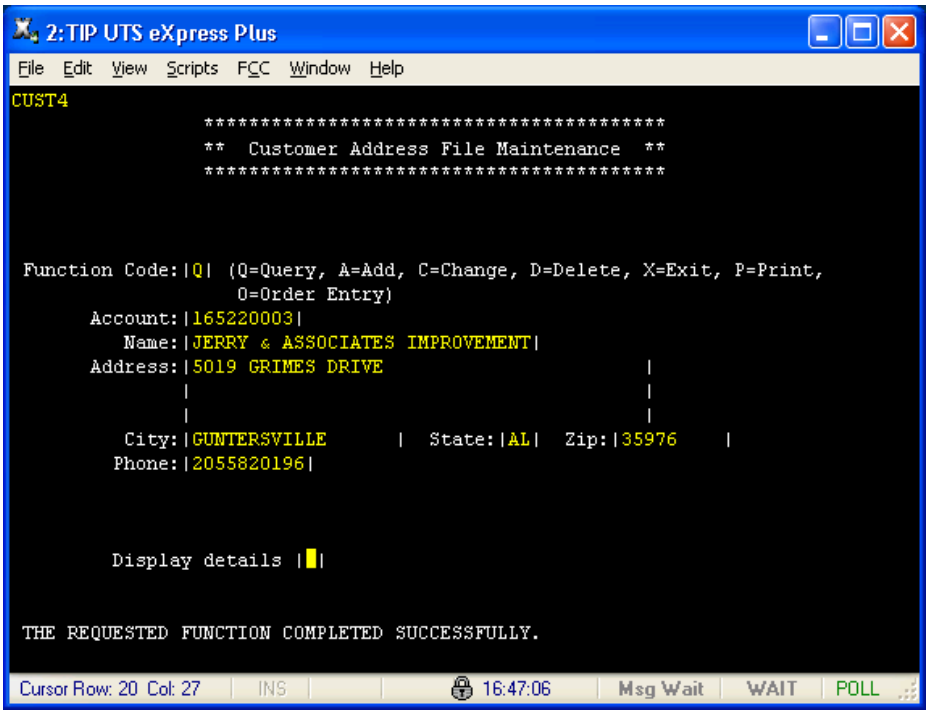
The following example is used to front-end a two-screen host TIP transaction application. To run this 2200 application (the old way) from a terminal or terminal emulator, the user performs the following tasks:

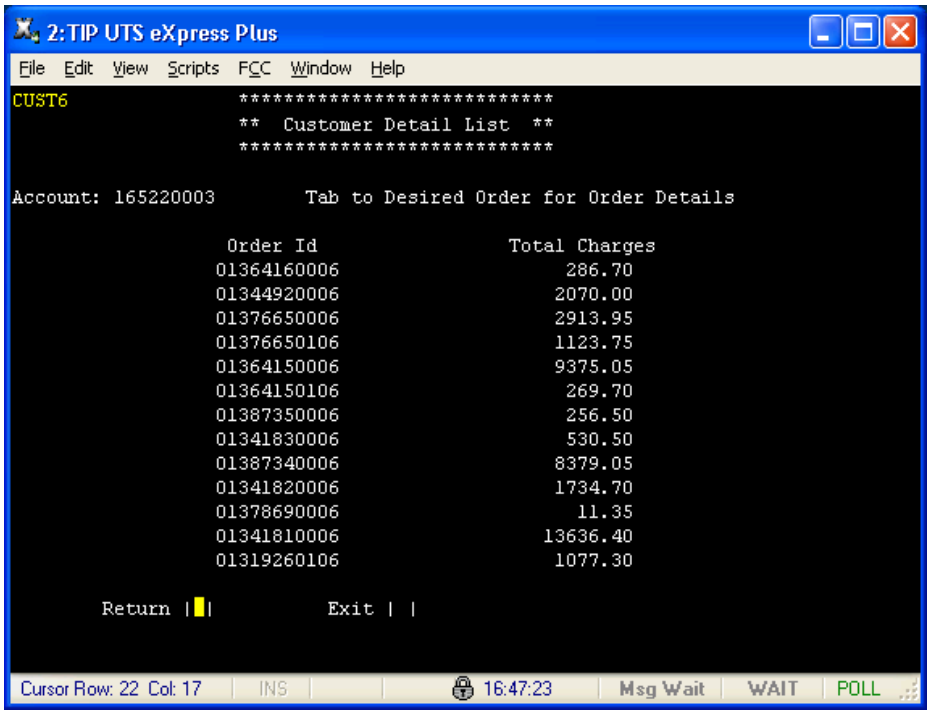
1. User establishes a host connection and signs on.
2. User enters the transaction code "**CUST3**" and transmits to get a blank **Customer Address File Maintenance** screen.
3. To get information for a customer, user enters "Q" in then **Function Code** field and a valid account number in the **Account** field, then transmits (see the results, next page).
4. Once customer name and address information is displayed, additional information can be obtained by moving the cursor to the **Display Details** field and transmitting.
5. To return to the **Customer Address File Maintenance** screen, the cursor is left in the Return field and the screen is transmitted.
6. To end the transaction sequence, an "X" is entered in the **Function Code** field and transmitted.

Note: This example is for a 2200 application, but the principal is the same if you were running an MCP application: once you sign on the MCP host, you perform some host process that paints information on the environment screen that you will eventually identify in the PC application that is to use the WinQT27x.ocx.

5.1 A Legacy Host Application

The legacy host application described on the previous page and run from a terminal emulator is shown below:





The **Customer Address File Maintenance** screen also supports **Add**, **Change** and **Delete** functions.

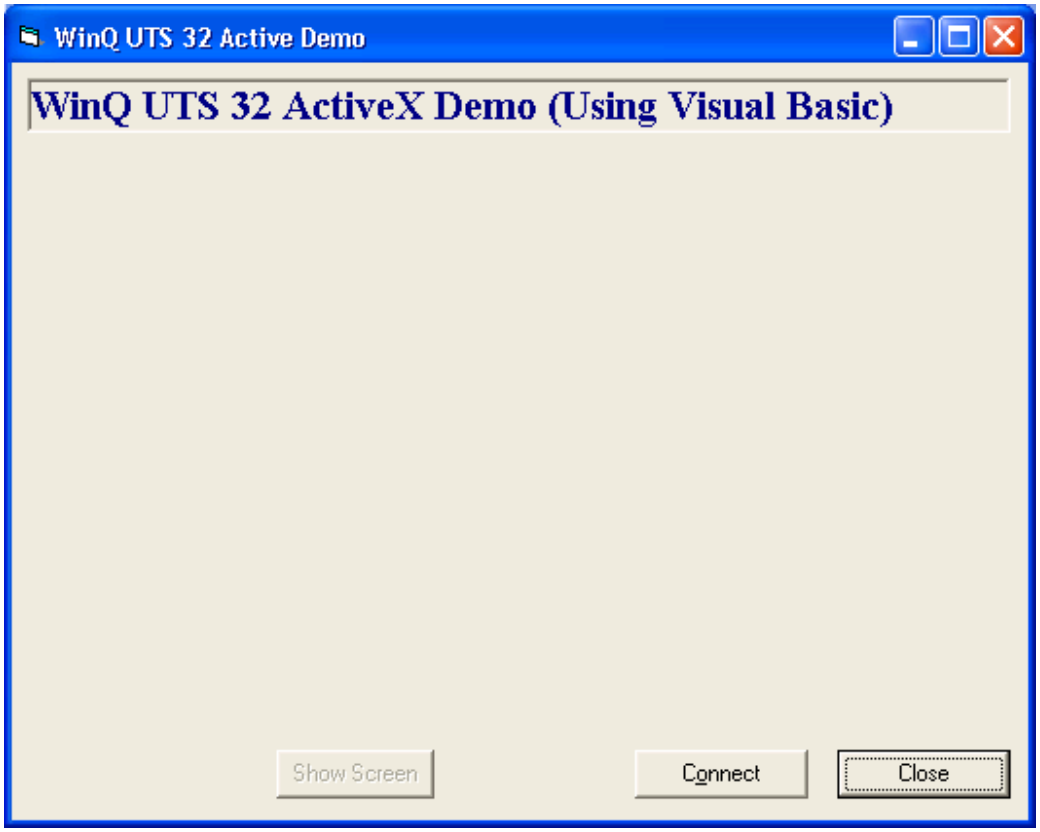
5.2 A Visual Basic Application

The WinQ Visual Basic application example shown here performs all the above functions with no screen visibility by the end user. All screen handling is done by the WinQ OCX component.

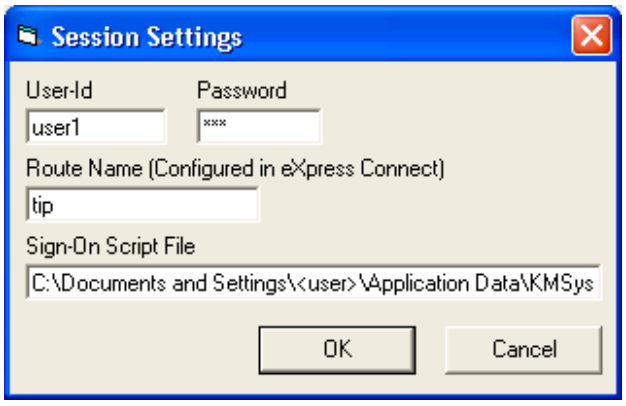
All Visual Basic Application Source code is included with eXpress Enterprise SDK and may be found in one of the installation directories after installation. Please check for the files in one of the following directories depending upon which product you install, T27 eXpress Enterprise or UTS eXpress Enterprise:

C:\Program Files\KMSYS Worldwide\T27 eXpress Enterprise\SDK\Samples
C:\Program Files\KMSYS Worldwide\UTS eXpress Enterprise\SDK\Samples

The initial main window displayed on startup contains no data entry areas because the user has not yet been validated. The user then clicks the **Connect** button to establish a host connection and sign on to the host application.

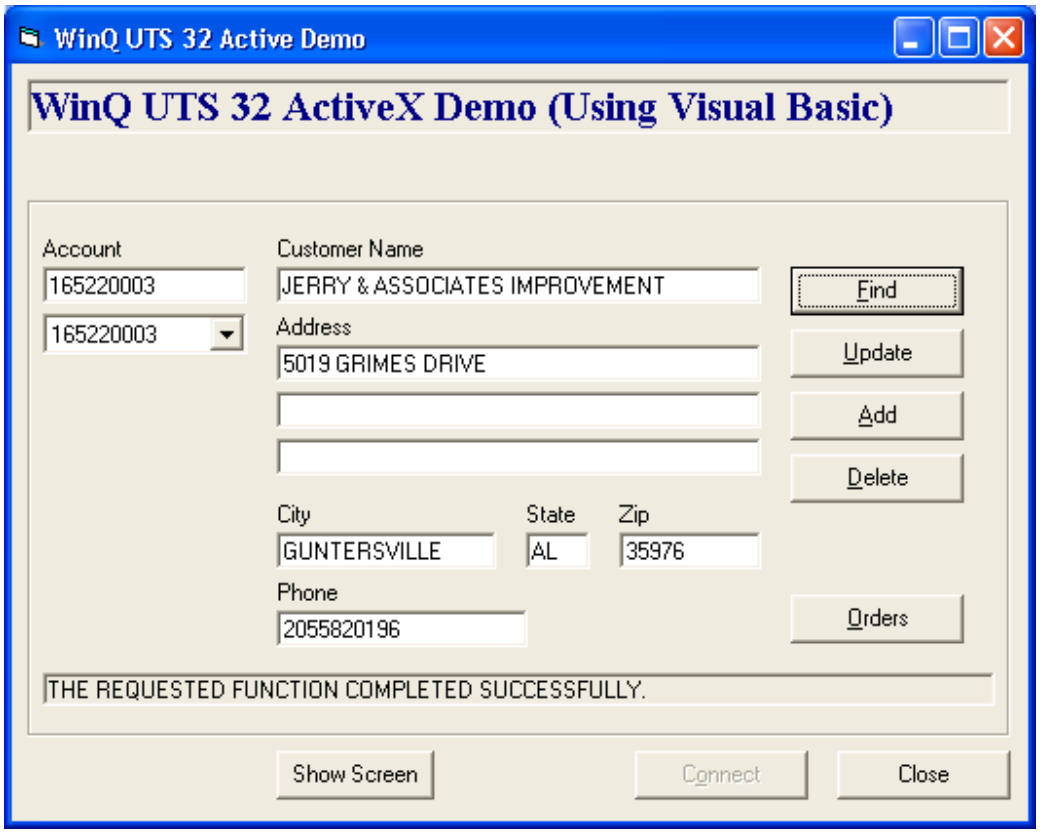


The Session Settings dialog is used to get user session settings needed for the connection and sign-on.



For this example, the **Session Settings** dialog contains more information than would normally be shown to an end user. It is recommended that only the user-id and password be solicited from the end user.

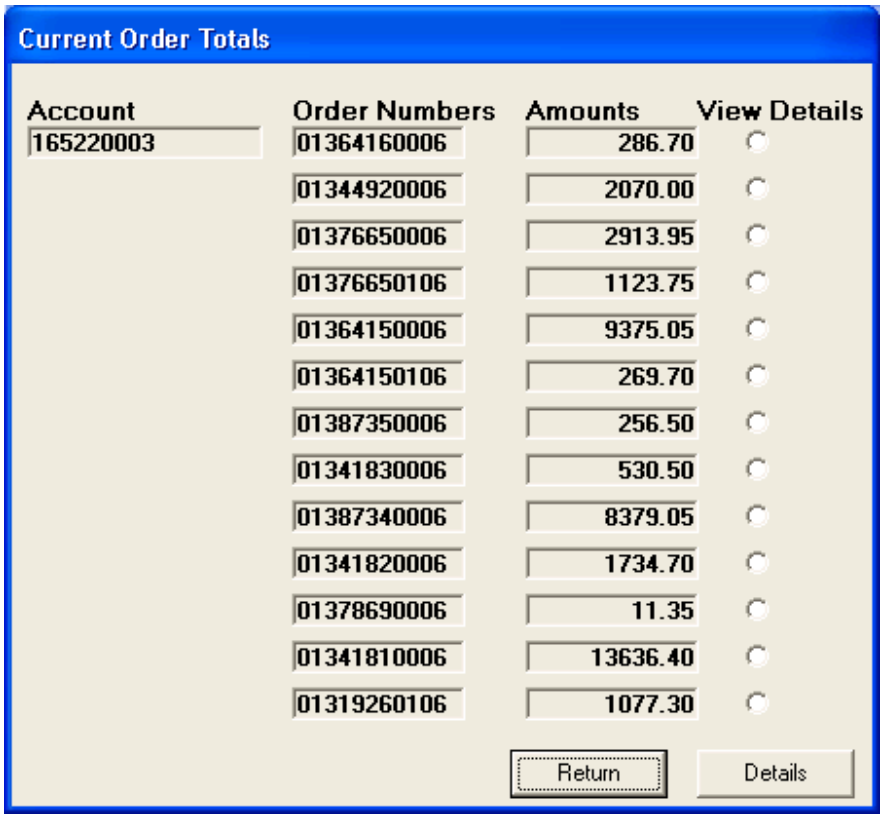
Once signed on, the main window is updated to show all other controls needed to run the application.



The combo box under the account number is preloaded with some valid account numbers for purposes of this example. When an account number in the combo box is clicked, the selected account is moved to the **Account** text box.

When the **Find**, **Update**, **Add** and **Delete** buttons are clicked, the program does what the user would have done if using the screen directly. That is, the appropriate function codes and data is moved to the internal screen and transmitted.

When the **Orders** button is clicked, the cursor is moved to the **Display Details** field in the internal screen and transmitted. When the host program displays the **Current Order Totals** window, the Windows application program displays the orders window.



Clicking **Return** transmits the internal screen with the cursor in the **Return** field.

Note: There is one more form (not shown) that used with this application and would be displayed if the user clicked an option button under the **View Details** column and clicked the **Details** button.

5.3 The Code for Each Control

The code shown in this section is executed when the user selects a specific control or when a form is loaded or unloaded. Note: Some of the lines shown have wrapped to additional lines due to the width limitation of this document. In addition, the source for the existing host application programs is not shown since the programs need not be changed, and perform exactly as they would on a dumb terminal. The Windows application program is the only part that uses the WinQ controls.

As was stated earlier, the first form (window) the user sees is **WinQDemo.frm**. When that form is loaded, the following code determines the location and size of the window and populates the combo box containing pre-selected account numbers:

```
Private Sub Form_Load()
    Left = (Screen.Width -Width) / 2
    Top = (Screen.Height -Height) / 2

    Cmb_Accounts.AddItem "215183000"
    Cmb_Accounts.AddItem "160090000"
    Cmb_Accounts.AddItem "160570001"
    Cmb_Accounts.AddItem "165220003"
    Cmb_Accounts.AddItem "167055000"
    Cmb_Accounts.AddItem "167055001"
    Cmb_Accounts.AddItem "168080000"

    WinQ32.ScreenVisible = 0 ' Hide screen
    WinQ32.WantScreenUpdated = 1 ' Set to process OnScreenUpdated events
End Sub
```

The first action the user takes is to press the **Connect** button, which causes the code associated with that event to be executed. That code initiates **SessionSettings.frm** and uses the **OpenSession** and **RunScript** methods to connect to the host. Note: The initial execution of the legacy host program is not performed through a WinQ control but by a eXpress script function. The **Connect** button code follows:

```
Private Sub Btn_Connect_Click()
Dim Rslt As Integer
    SessionSettings.Top = Top + (Height - SessionSettings.Height) / 2
    SessionSettings.Left = Left + (Width - SessionSettings.Width) / 2

    SessionSettings.Show (1)

    If SSResult = True Then
        MousePointer = vbHourglass
        Rslt = WinQ32.OpenSession(SessionSettings.Txt_Route.Text)
        If Rslt = 0 Then
            MsgBox "WinQ OpenSession Failed", mb_IconExclamation, "WinQ Error"
            MousePointer = vbDefault
            Exit Sub
        End If
        Rslt = WinQ32.RunScript(SessionSettings.Txt_Script, 2, Trim(SessionSettings.Txt_UserId),
Trim(SessionSettings.Txt_Password), "", "", "", "")
        If Rslt = 0 Then
            MsgBox "WinQ SignOn Script returned False", mb_IconExclamation, "WinQ SignOn Error"
MousePointer = vbDefault Exit Sub
        End If
        Frm_CustInfo.Visible = True
        Btn_Connect.Enabled = False
        Btn_Show.Enabled = True
        MousePointer = vbDefault
    End If
End Sub
```

The code for loading the **ScreenSettings** form is minimal since Visual Basic **Show** method was used to display the form:

```
Private Sub Form_Load()
    SSResult = False
End Sub
```

Only the public variable, **SSResult (Public SSResult As Boolean)**, is set to false indicating that sign-on has yet to take place.

The next event to take place is when the user enters information into the text boxes on the **SessionSettings** form. No code is associated with these controls.

When the user presses the **OK** button and if the connection was successful, the **SSResult** variable is set to true and the form is hidden from view:

```
Private Sub Btn_OK_Click()
    WinQDemo.SSResult = True
    Hide
End Sub
```

Also notice that upon successful completion of the sign-on process, the hidden frame control, **Frm_CustInfo**, is made visible (see **Btn_Connect_Click** code, above) revealing the controls allowing data entry for customer name and address information.

The eXpress Plus sign-on script, **WinQTIP.bas**, shown on the **Session Settings** dialog executes the host TIP transaction, **CUST3**:

```
' *** Script recorded by UTS eXpress Plus Script Recorder
Sub Main()
        dim UserId as String
        dim Password as String
        UserId = GetUserParam(1)
        Password = GetUserParam(2)
        ' Wait 10
    If not WaitForString(Chr$(31) & "7!M@" & Chr$(31) & "7`@@" & Chr$(27) & Chr$(11) & "7!" &
Chr$(15)) Then Exit Sub
    EnterText Chr$(30) & UserId & "/" & Password
    UTSKey UK_TRANSMIT_KEY
    ' Wait 10
    If not WaitForSpecificString(24, 1, Chr$(30) & "Enter your proj", 0) Then
        Exit Sub
    End If
    EnterText Chr$(30)
    UTSKey UK_TRANSMIT_KEY

    ' Wait 10
    If not WaitForSpecificString(24, 1, Chr$(30) & " ", 0) Then
        Exit Sub
    End If
    UTSKey UK_CURSOR_TO_HOME
    UTSKey UK_Erase_DISPLAY
    EnterText "CUST3 "
    UTSKey UK_TRANSMIT_KEY

    If not WaitForSpecificString(3, 18, "** Customer Address File Maintenance **", 0) Then
        Exit Sub
    End If
    Wait 1000
    SetSignOnResult 1
End Sub
```

As stated earlier, when the user selects and account from the combo box, the following code is executed to place that account in the **Account** text box:

```
Private Sub Cmb_Accounts_Click()
    Txt_Account = Cmb_Accounts.List(Cmb_Accounts.ListIndex)
End Sub
```

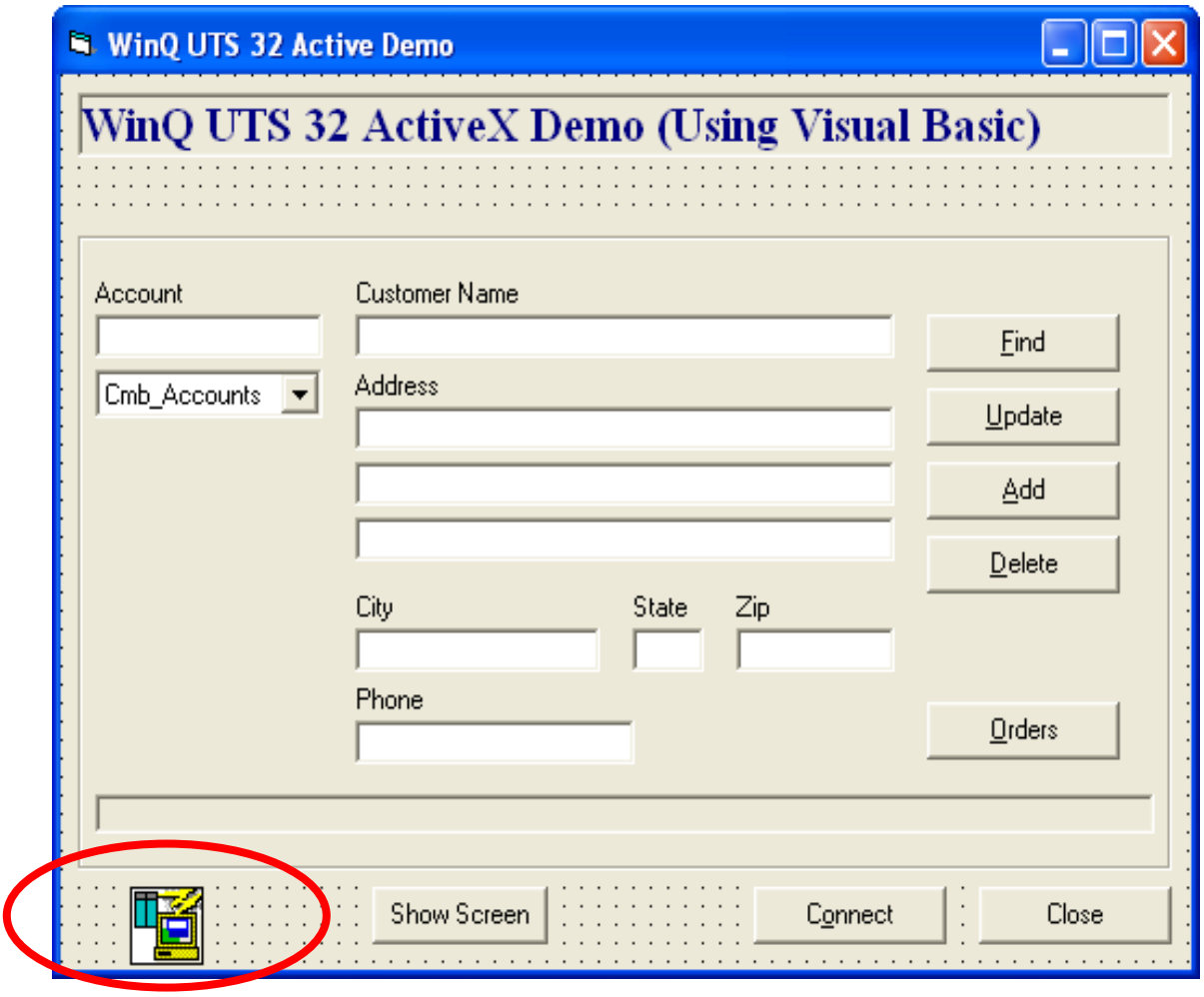
Next, the user presses the **Find** button. The code associated with the **Find** button, places a "Q" in row 8, column 17. This position is the location of the **Function Code** on the internal screen. Next, the selected account number is placed in the appropriate position, the cursor positioned to the next field after the account number, the internal screen refreshed and the transmit key is issued. The transaction code (**CUST4**) that was painted by the host program, **CUST3**, in row 1, column 1, executes the second transaction program that retrieves the data.

```
Private Sub Btn_Find_Click()
    Dim Status As Long
    Dim Rslt As Integer
    ' Enter the Q (Query) function code and Account (from Text box), then
    ' move the cursor to the next field and transmit.
    WinQ32.SetScreenText 8, 17, "Q"
    WinQ32.SetScreenText 10, 17, Txt_Account
    WinQ32.SetCursorPosition 11, 17

    WinQ32.RefreshScreen ' This is only to show screen changes
                         ' if it is visible.

    WinQ32.DoTerminalKey UK_TRANSMIT_KEY
End Sub
```

Upon receiving data from the host program, the **OnScreenUpdated** event examines what information was received from the host. This code is associated with the required WinQ Custom Control, **WinQUTS32x**, is executed as a result of the **WantScreenUpdated** property being set when the form was loaded.

```
Private Sub WinQ32_OnScreenUpdated()
    Dim X As Integer
    ' If the Order Total screen has been displayed by the host application
    ' ("CUST6" in upper left of screen), then show the Orders window with
    ' information from the Order Total host screen. Otherwize, assume the
    ' Customer Name and Address screen is still displayed and show
    ' information in the main window.
    If WinQ32.GetScreenText(1, 1, 5) = "CUST6" Then ' Orders Screen???
        Orders.Lbl_Account = Trim$(WinQ32.GetScreenText(5, 10, 9))
        For X = 0 To 12
            Orders.Lbl_Order(X) = Trim$(WinQ32.GetScreenText(8 + X, 18, 12))
            Orders.Lbl_Amt(X) = Trim$(WinQ32.GetScreenText(8 + X, 45, 12))
        Next X
        Orders.Show (1)
        Btn_Find_Click
    Else ' Assume Name and Address screen showing
        Txt_Name = RTrim$(WinQ32.GetScreenText(11, 17, 30))
        Txt_Addr1 = RTrim$(WinQ32.GetScreenText(12, 17, 40))
        Txt_Addr2 = RTrim$(WinQ32.GetScreenText(13, 17, 40))
        Txt_Addr3 = RTrim$(WinQ32.GetScreenText(14, 17, 40))
        Txt_City = RTrim$(WinQ32.GetScreenText(15, 17, 18))
        Txt_State = RTrim$(WinQ32.GetScreenText(15, 45, 2))
        Txt_Zip = RTrim$(WinQ32.GetScreenText(15, 55, 6))
        Txt_Phone = RTrim$(WinQ32.GetScreenText(16, 17, 10))
        Lbl_Msg = RTrim$(WinQ32.GetScreenText(23, 2, 78))
    End If
End Sub
```

The WinQ Custom Control is not visible to the user and, therefore, does not appear on the capture runtime image of the **WinQ UTS 32 ActiveX Demo** dialog shown earlier in this chapter. The following is the captured design image as shown in Visual Basic:



The control is in the bottom left-hand position of the form, but may be placed anywhere since it is not visible to the user. The **Visible** property for the control is set to **False**.

Assuming that there has been a successful retrieval of the customer account data, the user can press the **Orders** button, which positions the cursor in the **Detail** field of the internal screen and transmits:

```
Private Sub Btn_Orders_Click()
    ' To access the Order Screen, just move the cursor to the Details field
    ' in the screen and transmit.
    WinQ32.SetCursorPosition 20, 27
    WinQ32.DoTerminalKey UK_TRANSMIT_KEY
End Sub
```

Upon receiving a message from the host, the WinQ Custom Control code is once again executed to determine if the order detail information was returned from the host. If so, fields from the internal screen are moved to **Orders.frm** (see the code for **WinQ32_OnScreenUpdated**, above), the form is shown and positioned over the **WinQDemo** form:

```
Private Sub Form_Load()
    ' Position over main form
    Top = WinQDemo.Top
    Left = WinQDemo.Left
End Sub
```

When the user presses the **OK** button on the Orders window, the cursor is already positioned over the **Return** field on the internal screen; therefore, the code only has to transmit to execute the **CUST6** transaction that will return the **Customer Address File Maintenance** screen. Upon hiding the **Current Order Totals** window, the **OnScreenUpdated** event code will execute the **Btn_Find_Click** that will cause the previous customer name and address information to be retrieved again.

```
Private Sub Btn_OK_Click()
    ' Assume cursor is still positioned on the Return field in the
    ' screen, because user can not move it. Just transmitting will bring
    ' back the Customer Name and Address screen.
    WinQDemo.WinQ32.DoTerminalKey UK_TRANSMIT_KEY
    Hide
End Sub
```

The remaining code performs update, add and delete customers. The code for these controls follow:

```
Private Sub Btn_Update_Click()
    Dim Status As Long
    Dim Rslt As Integer

    ' Enter the C for Change function and move all data, execpt account,
    ' from text boxes to screen. Position the cursor to unlabled field
    ' following Phone Number and transmit.

    WinQ32.SetScreenText 8, 17, "C"
    WinQ32.SetCursorPosition 11, 17
    WinQ32.DoTerminalKey UK_ERASE_TO_END_DISPLAY
    WinQ32.SetScreenText 11, 17, Txt_Name
    WinQ32.SetScreenText 12, 17, Txt_Addr1
    WinQ32.SetScreenText 14, 17, Txt_Addr3
    WinQ32.SetScreenText 15, 17, Txt_City
    WinQ32.SetScreenText 15, 45, Txt_State
    WinQ32.SetScreenText 15, 55, Txt_Zip
    WinQ32.SetScreenText 16, 17, Txt_Phone
    WinQ32.SetCursorPosition 21, 73
    WinQ32.RefreshScreen ' This is only to show screen changes
                         ' if it is visible
    WinQ32.DoTerminalKey UK_TRANSMIT_KEY
End Sub

Private Sub Btn_Add_Click()
    Dim Status As Long
    Dim Rslt As Integer
    WinQ32.SetScreenText 8, 17, "A"
    WinQ32.SetCursorPosition 10, 17
    WinQ32.DoTerminalKey UK_ERASE_TO_END_DISPLAY
    WinQ32.SetScreenText 10, 17, Txt_Account
    WinQ32.SetScreenText 11, 17, Txt_Name
    WinQ32.SetScreenText 12, 17, Txt_Addr1
```

```
    WinQ32.SetScreenText 13, 17, Txt_Addr2
    WinQ32.SetScreenText 14, 17, Txt_Addr3
    WinQ32.SetScreenText 15, 17, Txt_City
    WinQ32.SetScreenText 15, 45, Txt_State
    WinQ32.SetScreenText 15, 55, Txt_Zip
    WinQ32.SetScreenText 16, 17, Txt_Phone
    WinQ32.SetCursorPosition 21, 73
    WinQ32.RefreshScreen ' This is only to show screen changes
                         ' if it is visible
    WinQ32.DoTerminalKey UK_TRANSMIT_KEY
End Sub

Private Sub Btn_Delete_Click()
    Dim Status As Long
    Dim Rslt As Integer
    If MsgBox("Are you sure you want to delete this account?", vbQuestion
Or vbYesNo, "Delete") = vbYes Then
        WinQ32.SetScreenText 8, 17, "D"
        WinQ32.SetCursorPosition 11, 17
        WinQ32.RefreshScreen ' This is only to show screen changes
                             ' if it is visible
        WinQ32.DoTerminalKey UK_TRANSMIT_KEY
    End If
End Sub
```

One other control, the **Show Screen** button, shows or hides the internal screen from the user. Where normally you would not want to display the internal screen to the user, a control of this type is very useful when debugging a new WinQ application.

```
Private Sub Btn_Show_Click()
    If Btn_Show.Tag = "" Then
        Btn_Show.Tag = "X"
        Btn_Show.Caption = "Hide Screen"
        WinQ32.ScreenVisible = True
    Else
        Btn_Show.Tag = ""
        Btn_Show.Caption = "Show Screen"
        WinQ32.ScreenVisible = False
    End If
End Sub
```

Finally, the user will close the application by pressing the **Close** button on the **WinQ UTS 32 Active Demo** form. The code simply uses the **CloseSession** method to break the connection to the host before terminating the application.

```
Private Sub Form_Unload(Cancel As Integer)
    WinQ32.CloseSession
End Sub
```

The complete code by form is shown below.

5.2.1 WinQDemo.frm

```
Public SSResult As Boolean
Private Sub Btn_Add_Click()
    Dim Status As Long
    Dim Rslt As Integer
    WinQ32.SetScreenText 8, 17, "A"
    WinQ32.SetCursorPosition 10, 17
    WinQ32.DoTerminalKey UK_ERASE_TO_END_DISPLAY
    WinQ32.SetScreenText 10, 17, Txt_Account
    WinQ32.SetScreenText 11, 17, Txt_Name
    WinQ32.SetScreenText 12, 17, Txt_Addr1
    WinQ32.SetScreenText 13, 17, Txt_Addr2
```

```
    WinQ32.SetScreenText 14, 17, Txt_Addr3
    WinQ32.SetScreenText 15, 17, Txt_City
    WinQ32.SetScreenText 15, 45, Txt_State
    WinQ32.SetScreenText 15, 55, Txt_Zip
    WinQ32.SetScreenText 16, 17, Txt_Phone
    WinQ32.SetCursorPosition 21, 73

    WinQ32.RefreshScreen ' This is only to show screen changes
                         ' if it is visible

    WinQ32.DoTerminalKey UK_TRANSMIT_KEY
End Sub

Private Sub Btn_Close_Click()
    WinQ32.CloseSession
    Unload Me
    End
End Sub

Private Sub Btn_Connect_Click()
    Dim Rslt As Integer
    SessionSettings.Top = Top + (Height - SessionSettings.Height) / 2
    SessionSettings.Left = Left + (Width - SessionSettings.Width) / 2

    SessionSettings.Show (1)

    If SSResult = True Then
        MousePointer = vbHourglass

        Rslt = WinQ32.OpenSession(SessionSettings.Txt_Route.Text)
        If Rslt = 0 Then
            MsgBox "WinQ OpenSession Failed", mb_IconExclamation, "WinQ Error"
            MousePointer = vbDefault
            Exit Sub
        End If
        Rslt = WinQ32.RunScript(SessionSettings.Txt_Script, 2, Trim(SessionSettings.Txt_UserId),
Trim(SessionSettings.Txt_Password), "", "", "", "")

        If Rslt = 0 Then
            MsgBox "WinQ SignOn Script returned False", mb_IconExclamation,
"WinQ SignOn Error"
            MousePointer = vbDefault
            Exit Sub
        End If
        Frm_CustInfo.Visible = True
        Btn_Connect.Enabled = False
        Btn_Show.Enabled = True
        MousePointer = vbDefault
    End If
End Sub

Private Sub Btn_Delete_Click()
    Dim Status As Long
    Dim Rslt As Integer

    If MsgBox("Are you sure you want to delete this account?", vbQuestion
Or vbYesNo, "Delete") = vbYes Then
        WinQ32.SetScreenText 8, 17, "D"
        WinQ32.SetCursorPosition 11, 17

        WinQ32.RefreshScreen ' This is only to show screen changes
                             ' if it is visible

        WinQ32.DoTerminalKey UK_TRANSMIT_KEY
    End If
```

```
End Sub

Private Sub Btn_Find_Click()
    Dim Status As Long
    Dim Rslt As Integer
    ' Enter the Q (Query) function code and Account (from Text box), then
    ' move the cursor to the next field and transmit.
    WinQ32.SetScreenText 8, 17, "Q"
    WinQ32.SetScreenText 10, 17, Txt_Account
    WinQ32.SetCursorPosition 11, 17
    WinQ32.RefreshScreen ' This is only to show screen changes
                         ' if it is visible

    WinQ32.DoTerminalKey UK_TRANSMIT_KEY
End Sub

Private Sub Btn_Orders_Click()
    ' To access the Order Screen, just move the cursor to the Details field
    ' in the screen and transmit.
    WinQ32.SetCursorPosition 20, 27
    WinQ32.DoTerminalKey UK_TRANSMIT_KEY
End Sub

Private Sub Btn_Show_Click()
    If Btn_Show.Tag = "" Then
        Btn_Show.Tag = "X"
        Btn_Show.Caption = "Hide Screen"
        WinQ32.ScreenVisible = True

    Else
        Btn_Show.Tag = ""
        Btn_Show.Caption = "Show Screen"
        WinQ32.ScreenVisible = False
    End If
End Sub

Private Sub Btn_Update_Click()
    Dim Status As Long
    Dim Rslt As Integer
    ' Enter the C for Change function and move all data, execpt account,
    ' from text boxes to screen. Position the cursor to unlabled field
    ' following Phone Number and transmit.
    WinQ32.SetScreenText 8, 17, "C"
    WinQ32.SetCursorPosition 11, 17
    WinQ32.DoTerminalKey UK_ERASE_TO_END_DISPLAY
    WinQ32.SetScreenText 11, 17, Txt_Name
    WinQ32.SetScreenText 12, 17, Txt_Addr1
    WinQ32.SetScreenText 13, 17, Txt_Addr2
    WinQ32.SetScreenText 14, 17, Txt_Addr3
    WinQ32.SetScreenText 15, 17, Txt_City
    WinQ32.SetScreenText 15, 45, Txt_State
    WinQ32.SetScreenText 15, 55, Txt_Zip
    WinQ32.SetScreenText 16, 17, Txt_Phone
    WinQ32.SetCursorPosition 21, 73

    WinQ32.RefreshScreen ' This is only to show screen changes
                         ' if it is visible

    WinQ32.DoTerminalKey UK_TRANSMIT_KEY
End Sub

Private Sub Cmb_Accounts_Click()
    Txt_Account = Cmb_Accounts.List(Cmb_Accounts.ListIndex)
End Sub

Private Sub Form_Load()
```

```
    Left = (Screen.Width -Width) / 2
    Top = (Screen.Height -Height) / 2
    Cmb_Accounts.AddItem "215183000"
    Cmb_Accounts.AddItem "160090000"
    Cmb_Accounts.AddItem "160570001"
    Cmb_Accounts.AddItem "165220003"
    Cmb_Accounts.AddItem "167055000"
    Cmb_Accounts.AddItem "167055001"
    Cmb_Accounts.AddItem "168080000"

    WinQ32.ScreenVisible = 0
    WinQ32.WantScreenUpdated = 1
End Sub

Private Sub Form_Unload(Cancel As Integer)
    WinQ32.CloseSession
End Sub

Private Sub WinQ32_OnScreenUpdated()
    Dim X As Integer

    ' If the Order Total screen has been displayed by the host application
    ' ("CUST6" in upper left of screen, then show the Orders window with
    ' information from the Order Total host screen. Otherwize, assume the
    ' Customer Name and Address screen isstill displayed and show
    ' information in the main window.

    If WinQ32.GetScreenText(1, 1, 5) = "CUST6" Then ' Orders Screen???
        Orders.Lbl_Account = Trim$(WinQ32.GetScreenText(5, 10, 9))
        For X = 0 To 12
            Orders.Lbl_Order(X) = Trim$(WinQ32.GetScreenText(8 + X, 18, 12))
            Orders.Lbl_Amt(X) = Trim$(WinQ32.GetScreenText(8 + X, 45, 12))
        Next X
        Orders.Show (1)
        Btn_Find_Click
    Else ' Assume Name and Address screen showing
        Txt_Name = RTrim$(WinQ32.GetScreenText(11, 17, 30))
        Txt_Addr1 = RTrim$(WinQ32.GetScreenText(12, 17, 40))
        Txt_Addr2 = RTrim$(WinQ32.GetScreenText(13, 17, 40))
        Txt_Addr3 = RTrim$(WinQ32.GetScreenText(14, 17, 40))
        Txt_City = RTrim$(WinQ32.GetScreenText(15, 17, 18))
        Txt_State = RTrim$(WinQ32.GetScreenText(15, 45, 2))
        Txt_Zip = RTrim$(WinQ32.GetScreenText(15, 55, 6))
        Txt_Phone = RTrim$(WinQ32.GetScreenText(16, 17, 10))
        Lbl_Msg = RTrim$(WinQ32.GetScreenText(23, 2, 78))
    End If
End Sub
```

5.2.2 SessionSettins.frm

```
Private Sub Btn_Cancel_Click()
    WinQDemo.SSResult = False Hide
End Sub

Private Sub Btn_OK_Click()
    WinQDemo.SSResult = True Hide
End Sub

Private Sub Form_Load()
    SSResult = False
End Sub
```

5.2.3 Orders.frm

```
Option Explicit
Private Sub Btn_OK_Click()
    ' Assume cursor is still positioned on the Return field in the
    ' screen, because user can not move it.
    ' Just transmitting will bring back the Customer Name and Address
    ' screen.
    WinQDemo.WinQ32.DoTerminalKey UK_TRANSMIT_KEY
    Hide

End Sub

Private Sub Form_Load()
    ' Position over main form
    Top = WinQDemo.Top
    Left = WinQDemo.Left
End Sub
```

Chapter 6: Application Operation Mode Procedures

The Application Operation Mode (AOM) on the host side is a very easy-to-use subroutine library that is called whenever the host application is required to communicate with the Windows application. There are only four WinQ functions used by the host application. The four functions are:

- Initialize WinQ environment.
- Receive Message (command) from the PC application.
- Send Message (response) to PC application.
- Terminate WinQ environment (release resources).

There are no other terminal I/O or screen management functions performed by the host application program.

The Windows side of the application is also very simple, but contains some additional functions since it is considered the client, or controlling, portion of the application. WinQ provides the following functions:

- Open and Close the session.
- Process Sign-On/Sign-Off - Process application's session establishment script.
- Send Message - Send text message (command) to the host application.
- Receive Message - Get a message sent from the host application.
- Send File - Send files to the host for use in the host application.
- Receive File - Receive files from the host for use in the Windows Application.

Developing a WinQ AOM application is quite different from developing a DPS2200 application. In the WinQ AOM environment, you are not limited by screen size, and you do not have to write complicated screen management routines and complex routines to handle display of data on screens. With WinQ AOM, a minimum amount of data formatting is required in the host application — just enough to present it to the PC/Workstation application. In addition, large quantities of data can be presented to the user at one time via file transfer functions, instead of having to manage multiple 24x80 screen displays.

The Windows side of the WinQ AOM application is very powerful, yet still simple compared to a DPS application. Presentation of large quantities of data to the user can be handled using Windows controls such as scrolling lists or sophisticated spreadsheet controls. Many Windows development tools make complex screen (window) format development very easy. Let us look at a typical inquiry application scenario:

- The user starts the Windows application by clicking the appropriate icon, just like any other Windows application.
- The application then calls the WinQ functions to connect to the host and process the user's session sign-on script for this application. This call automatically begins communications with the host system. Once communication is established, the user is automatically signed on and the host application program is initiated. Since the host application program is playing the roll of the server, it simply calls the WinQ Receive Message function and waits for a command from the Windows application.
- Once the sign-on is completed, the WinQ Windows application displays a window to solicit input from the user. When the WinQ Windows application determines what the user wants, it sends a command in the form of a text message to the host program via the WinQ Send Message function to tell the host program what it needs.
- The host application receives the message from the WinQ Windows application and processes it accordingly. When complete, the host program sends a response message to the WinQ Windows application telling it whether the host program was successful. The host application then awaits its next command from the Windows application.
- Upon receipt of the host response message, the WinQ Windows program may do several things. If the host command is, for example, simply to determine if a record exists, or to get information that can be contained in a single text message, the Windows program can parse the host message and display a window containing the result for the user. On the other hand, if the request is for a large quantity of information, the applications designer may choose to have the host program extract and format the data and write it to a sequential transfer file. The initial response message from the host program may contain only the number of records written to the file. In this case, the Windows program will then call the WinQ Get File function to transfer the sequential file from the host to the PC automatically. Upon returning from the Get File function, the data can be read and displayed in a scrolling text window. When a file is being transferred to or from the host, the WinQ API automatically performs the file transfer for the application. When complete, the WinQ Windows program is notified. The host program is not aware of the file transfer activity. The WinQ Windows program continues its conversation with the user.

The file transfer functions of WinQ are one of its most important features. WinQ handles file transfer automatically. The application programs only need to read or write the file — no communications handling is required by either the host or Windows application programs. WinQ file transfer is also very efficient, utilizing a very powerful data compression scheme and handling automatic error detection and recovery.

Chapter 7: Application Considerations

This section will discuss several things to be considered when developing an application using the WinQ Application Operation Method (2200 only).

7.1 Data Formats

The following considerations deal with data format and contents of WinQ messages.

7.1.1 No Binary Data in WinQ Messages

No binary data is allowed within WinQ messages. There are two reasons for not allowing binary data to be sent in WinQ messages:

1. Often binary data is interpreted by the communications transport as special control character sequences, which can cause loss of data between the client and server programs. Additionally WinQ Converts all characters outside the normal ASCII character set to spaces.
2. Binary formats between the PC/Workstation and the host system may be incompatible. For instance, the 2200 byte is made up of nine (9) bits and the Intel-based PC byte consists of eight (8) bits; thus, one bit out of nine sent from a 2200 host would be lost.

7.1.2 Files Used in File Transfer Operations

WinQ currently supports only host 2200 file types of SDF, PCIOS or symbolic elements in program files containing only ASCII data. On the PC/Workstation side, only ASCII text files are supported.

7.2 Timeouts

Most WinQ functions provide a timeout value. The timeout value specifies how long the function should wait (in seconds) before returning an error. Use reasonable timeout values in your application, taking into consideration the operation to be performed.

If a timeout error occurs, the WinQ operation may still be in process. You may need to resynchronize you conversation. You may resynchronize by first performing a **InterruptOperation** function. If the **InterruptOperation** is not successful, you may have lost your communications link to the host. If the **InterruptOperation** is successful, you should send a message to the host server telling it to respond with a synchronization message.

7.3 User Interface

The user interface design of any Windows application is very important. Consider the following:

7.3.1 Keep User from Initiating Multiple WinQ Functions

Because your Windows application is asynchronous and event driven, it is possible for the user to initiate several functions while other functions are in process. To prevent inadvertent execution of multiple WinQ functions, disable any controls that would allow the user to initialize another WinQ function.

7.3.2 Keep the User Informed

Since some WinQ application activities can take a long time to complete (e.g., a large file transfer), it is a good idea to tell the user what is happening. You can use status bar displays, or even animated controls to show the user that the application is still processing.

7.4 Host Application

Since a WinQ application is typically conversational, be careful not to lock up the host database too long — use IMPART/DEPART or BEGIN/END THREAD around each logical transaction.

Chapter 8: WinQ Host Subroutine Library

When using the WinQ Application Operation Mode, a subroutine library is provided that may be installed locally on the 2200 host. The subroutine library contains all function necessary for the host program to communicate with the Windows program. There are only four WinQ functions used in the WinQ 2200 subroutine interface: **WINQ_SVRINIT**, **WINQ_SVRGET**, **WINQ_SVRPUT** and **WINQ_SVRTERM**.

8.1 ASCII COBOL WinQ API

8.1.1 Initialize WinQ Environment

WINQ_SVRINIT is used to initialize the WinQ Application interface environment and must be called, only once, prior to calling any other WinQ function.

Format:

ENTER MASM 'WINQ_SVRINIT'.

8.1.2 Receive (Get) a Message Sent from the Windows Application

WINQ_SVRGET is used to receive messages from the WinQ Windows application. Program execution will be halted until a message is received from the WinQ Windows PC application, or an error is detected within the WinQ API.

Format:

ENTER MASM 'WINQ_SVRGET' USING *Status*, *Msg-buffer*, *Msg-length*.

Status is a COBOL 01 or 77 level defined as PIC S9(10) COMPUTATIONAL. Status will be less than, or equal to, zero upon normal completion. Any value greater that zero is to be considered a WinQ interface error.

Msg-buffer is a COBOL 01 level data item used for receiving messages from the Windows application.

Msg-length is a COBOL 01 or 77 level defined as PIC S9(10) COMPUTATION. *Msg-length* will contain the length of the message received.

8.1.3 Send (Put) a Message to the Windows PC Application

The **WINQ_SVRPUT** is used to send a single message to the WinQ Windows Application. Messages may consist of up to 4000 characters of ASCII text. A message should not contain any control characters (ASCII values less than 20 or greater than 128). Any control characters found in the message will be converted to spaces prior to transmission.

Format:

ENTER MASM 'WINQ_SVRPUT' USING *Status*, *Msg-buffer*, *Msg_length*.

Status is a COBOL 01 or 77 level defined as PIC S9(10) COMPUTATIONAL. Status will be less than, or equal to, zero upon normal completion. Any value greater that zero is to be considered a WinQ interface error.

Msg-buffer is a COBOL 01 level data items used for receiving messages from the Windows application.

Msg-length is a COBOL 01 or 77 level defined as PIC S9(10) COMPUTATIONAL. *Msg-length* must contain the length of the message to be sent to the Windows application.

8.1.4 Terminate the WinQ Interface

The **WINQ_SVRTERM** function is used to release all WinQ resources. It does not terminate your program. No further WinQ API call can be made unless **WINQ_SVRINIT** is called again.

Format:

ENTER MASM 'WINQ_SVRTERM'.

Chapter 9: Application Operation Mode Example

This section illustrates a complete WinQ Application Operation Mode example, including both the application and host programs. The application used in this sample will contain very simple logic so as not to confuse WinQ development issues with application logic.

9.1 The Sample Application

For this example, a simple name and address file maintenance application was developed. The application will provide functions, to add, change and delete, name and address records stored in a DMS 2200 database. The application also provides the ability to search for records based on state, city and zip code. The client or PC program is written in Microsoft's Visual Basic and the host server program is written in ASCII COBOL (@ACOB).

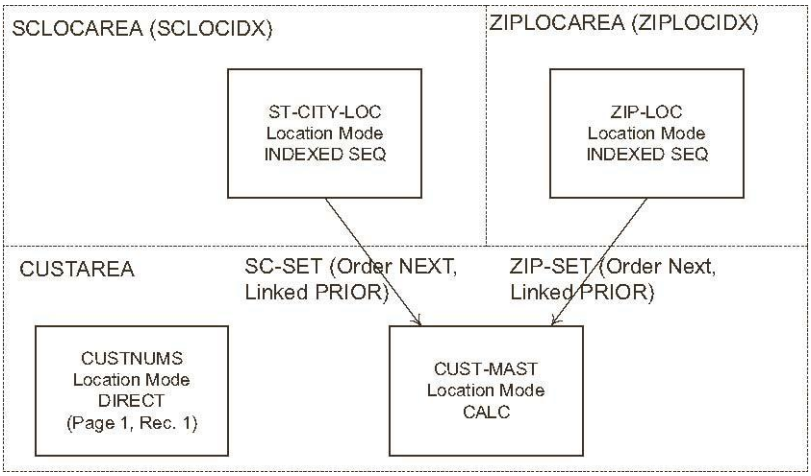
The complete source code for both the PC and host programs are installed in the UTS eXpress Enterprise directory after installation:

C:\Program Files\KMSYS Worldwide\UTS eXpress Enterprise\SDK\Samples

Note: In this example, the background screen is displayed and a trace option is set as would be commonly used when developing and debugging the application. These options are easily reset prior to production implementation. See **ScreenVisible** (pages 4-3 and 9-17) and **TraceOption** (pages 4-4 and 9-13).

9.1.1 The Database Structure

The DMS 2200 database structure used for this application was designed to provide ease of access as well as ease of maintenance. The following block diagram shows the various record relationships:



See the schema listing in Chapter 10 for the detail record layouts.

9.1.2 The Host Server Program

The host server program, **CUSTADDR.COB** (WINQCOB), contains a command loop, which is used to execute various commands sent for the WinQ Client program to process. Each command that requires database access is treated like a separate transaction beginning with an IMPART and ending with a DEPART.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. WINQCOB.
 DATE-COMPILED.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. UNIVAC-2200-200.
 OBJECT-COMPUTER. UNISYS-2200-200 MEMORY 3 MODULES.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT WORKFILE ASSIGN TO DISC WORKFILE.
 DATA DIVISION.
 SUBSCHEMA SECTION.
 INVOKE SUBSCHEMA CUSTSUB IN FILE SCHEMA-ABS
     OF SCHEMA CUSTSCHEMA
     COPYING RECORDS INTO WORKING
     COPYING DATA-NAMES INTO WORKING
     ERROR RECOVERY IS 9000-GEN-ERROR
     ROLLBACK IS 9100-ROLL-BACK.
 FILE SECTION.
 FD WORKFILE
     LABEL RECORDS ARE OMITTED
    DATA RECORD IS WORKREC.
 01 WORKREC    PIC X(300).
 WORKING-STORAGE SECTION.
*** INPUT CMD-ACTION DEFINITION.
 01 CMD-ACTION.
     05 CM-OP  PIC XX.
     05 CM-DATA       PIC X(300).
*** WINQ MESSAGE DEFINITION.
```

```
 01 WINQ-STAT  PIC 9(10) COMP.
 01 WINQ-LEN   PIC 9(10) COMP.
 01 WINQ-BUFF.
     05 OB-HEAD        PIC X(5).
     05 OB-BLOCK       PIC X(3000).
 01 PTR        PIC 9(10) COMP.
 01 SEARCH-STATE-CITY.
     05 SEARCH-STATE   PIC XX.
     05 SEARCH-CITY    PIC X(20).
 01 SEARCH-ZIP         PIC X(9).
 01 SEARCH-NAME        PIC X(20).
* CUST-ADDR RECORD WORK AREA (ALL ASCII DATA)
* THIS AREA IS USED TO FORMAT COMPLETE CUST MAST RECORDES
* FOR EXCHANGE WITH THE CLIENT APPLICATION.
* CUST SELECTION LIST RECORD AREA.
 01 WORK-AREA.
     05 WRK-CUSTNUM    PIC 9(10).
     05 WRK-LAST       PIC X(20).
     05 WRK-FIRST      PIC X(20).
     05 WRK-MI PIC X.
     05 WRK-COMPANY    PIC X(30).
     05 WRK-ADDR1      PIC X(30).
     05 WRK-ADDR2      PIC X(30).
     05 WRK-ADDR3      PIC X(30).
     05 WRK-CITY       PIC X(20).
     05 WRK-STATE      PIC XX.
     05 WRK-ZIP        PIC X(10).
     05 WRK-ACODE      PIC X(3).
     05 WRK-PHONE      PIC X(7).

* THIS AREA IS USED TO SEND LISTS OF SELECTED CUSTOMERS
* TO THE CLIENT APPLICATION.
 01 LIST-AREA.
     05 WRK EXT        PIC X(5).
     05 LA-CUSTNUM     PIC 9(10).
     05 LA-LAST        PIC X(20).
     05 LA-FIRST       PIC X(20).
     05 LA-MI          PIC X.
     05 LA-CITY        PIC X(20).
     05 LA-STATE       PIC XX.
     05 LA-ZIP         PIC X(10).
* THE FOLLOWING ARE USED FOR DYNAMICALLY ASSIGNING
* A TEMP WORK FILE FOR USE IN DOWNLOADS.
 01 CSF-IMG            PIC X(80).
 01 CSF-STATUS         PIC 9(10) COMP.

 01 CUST-NUM-A                PIC X(10).
 01 CUST-NUM-N REDEFINES CUST-NUM-A PIC 9(10).

 PROCEDURE DIVISION.
 0000-MAIN SECTION.
 0100-MS.

     PERFORM 1000-INIT.
     MOVE 'OK000' TO OB-HEAD.
     MOVE 5 TO WINQ-LEN.
     PERFORM 8200-WINQ-PUT.

*** CMD-ACTION PROCESSING LOOP.
 0200-MS.
     PERFORM 8100-WINQ-GET.
*** EXIT APPLICATION COMMAND.
     IF CM-OP = 'XX'
         GO TO 0900-DONE.
```

```
*** RETRIEVE ADDRESS RECORD BY CUST NUMBER

     IF CM-OP = 'RT'
         PERFORM 2000-RET-ADDR
         GO TO 0200-MS.

*** REPLACE ADDRESS RECORD BY CUST NUMBER
     IF CM-OP = 'RP'
         PERFORM 2100-REP-ADDR
         GO TO 0200-MS.

*** ADD NEW ADDRESS RECORD
     IF CM-OP = 'AD'
         PERFORM 2200-ADD-ADDR
         GO TO 0200-MS.

*** DELETE ADDRESS RECORD BY CUST NUM
     IF CM-OP = 'DT'
         PERFORM 2300-DEL-ADDR
         GO TO 0200-MS.

*** RETRIEVE LIST OF RECORDS BY STATE/CITY SEARCH
     IF CM-OP = 'SS'
         PERFORM 2400-SEARCH-STATE
         GO TO 0200-MS.
*** RETRIEVE LIST OF RECORDS BY ZIP SEARCH
     IF CM-OP = 'SZ'
         PERFORM 2500-SEARCH-ZIP
         GO TO 0200-MS.

*** CREATE A FILE CONTAINING ALL RECORDS FOR DOWNLOAD
     IF CM-OP = 'DL'
         PERFORM 2600-DOWNLOAD-FILE
         GO TO 0200-MS.

*** RESPOND TO ACKNOWLEDGE APPLICATION IS STILL CONNECTED
     IF CM-OP = 'WQ'
         MOVE 'K000'TO OB-HEAD
         MOVE 5 TO WINQ-LEN
         PERFORM 8200-WINQ-PUT
         GO TO 0200-MS.

*** UNKNOWN COMMAND.
     MOVE SPACES TO WINQ-BUFF.
     MOVE 'R999' TO OB-HEAD.
     MOVE CM-DATA TO OB-HEAD.
     MOVE 30 TO WINQ-LEN.
     PERFORM 8200-WINQ-PUT.
     GO TO 0200-MS.
 0900-DONE.
     MOVE 'OK000' TO OB-HEAD.
     MOVE 5 TO WINQ-LEN.
     PERFORM 8200-WINQ-PUT.

*** TERMINATE WINQ API ***
     ENTER MASM 'WINQ_SVRTERM'.
     STOP RUN.
 0999-EXIT.
     EXIT.

 1000-INIT SECTION.
 1100-I.
*** INITIALIZE WINQ HOST API ***
     ENTER MASM 'WINQ_SVRINIT'.
*** ASSIGN TEMP FILE FOR DOWNLOAD, IF REQUESTED.
     MOVE '@ASG,T WORKFILE.,F . ' TO CSF-IMG.
     MOVE 0 TO CSF-STATUS.
```

```
     CALL 'ERACSF$' USING CSF-IMG, CSF-STATUS.
     IF CSF-STATUS < 0
         MOVE 'ER999 CAN NOT ASSIGN TEMP WORK FILE.'
             TO WINQ-BUFF
         MOVE 40 TO WINQ-LEN
         PERFORM 8200-WINQ-PUT
         STOP RUN.

 1199-EXIT.
     EXIT.

*****************************************************
* RETRIEVE ONE ADDRESS RECORD BY CUSTOMER NUMBER
*****************************************************
 2000-RET-ADDR SECTION.
 2001-RA.
     IMPART.
     OPEN CUSTAREA USAGE-MODE IS RETRIEVAL.
     MOVE 'CUSTAREA' TO CUST-ANAME.
     MOVE CM-DATA TO CUST-NUM-A.
     MOVE CUST-NUM-N TO CM-CUSTNUM.
     FETCH CUST-MAST RECORD
          ON ERROR GO TO 2080-RA.
     MOVE 'OK000' TO OB-HEAD.
     MOVE 300 TO WINQ-LEN.
     PERFORM 4100-MOVE-TO-WORK.
     MOVE WORK-AREA TO OB-BLOCK.
     DEPART.
     PERFORM 8200-WINQ-PUT.
     GO TO 2099-EXIT.
 2080-RA.
     IF ERROR-NUM = '0013'
         MOVE 'OK999' TO WINQ-BUFF
         MOVE 5 TO WINQ-LEN
         DEPART
         PERFORM 8200-WINQ-PUT
     ELSE
         GO TO 9000-GEN-ERROR.
 2099-EXIT.
     EXIT.

*****************************************************
* REPLACE ONE ADDRESS RECORD BY CUSTOMER NUMBER
*****************************************************
 2100-REP-ADDR SECTION.
 2101-RA.
     MOVE CM-DATA TO WORK-AREA.
     IMPART.
     OPEN CUSTAREA USAGE-MODE IS UPDATE.
     MOVE 'CUSTAREA' TO CUST-ANAME.
     MOVE CM-DATA TO WORK-AREA.
     MOVE WRK-CUSTNUM TO CM-CUSTNUM.

* FETCH RECORD TO BE REPLACED.
     FETCH CUST-MAST RECORD
         ON ERROR GO TO 2180-RA.
* IF STATE, CITY OR ZIP HAS CHANGED, DELETE RECORD
* AND STORE BACK UNDER NEW LOCATORS.

     IF CM-STATE NOT = WRK-STATE
     OR CM-CITY NOT = WRK-CITY
     OR CM-ZIP NOT = WRK-ZIP
         DELETE CUST-MAST RECORD
         PERFORM 4200-MOVE-TO-REC
         PERFORM 3000-GET-LOCATORS
         STORE CUST-MAST
```

```
     ELSE
* REPLACE RECORD DATA AND MODIFY EXISTING RECORD.
     PERFORM 4200-MOVE-TO-REC
     MODIFY CUST-MAST RECORD.
     MOVE 'OK000' TO OB-HEAD.
     MOVE 5 TO WINQ-LEN.
     DEPART.
     PERFORM 8200-WINQ-PUT.
     GO TO 2199-EXIT.

 2180-RA.
     IF ERROR-NUM = '0013'
         MOVE 'OK999' TO WINQ-BUFF
         MOVE 5 TO WINQ-LEN
         DEPART WITH ROLLBACK
         PERFORM 8200-WINQ-PUT
     ELSE
         GO TO 9000-GEN-ERROR.
 2199-EXIT.
     EXIT.

*****************************************************
* ADD AN ADDRESS RECORD BY CUSTOMER NUMBER
*****************************************************
 2200-ADD-ADDR SECTION.
 2201-AA.
     MOVE CM-DATA TO WORK-AREA.
     IMPART.
     OPEN CUSTAREA USAGE-MODE IS UPDATE.
     OPEN SCLOCIDX USAGE-MODE IS RETRIEVAL.
     OPEN ZIPLOCIDX USAGE-MODE IS RETRIEVAL.
     OPEN SCLOCAREA USAGE-MODE IS UPDATE.
     OPEN ZIPLOCAREA USAGE-MODE IS UPDATE.
     MOVE 'CUSTAREA' TO CUST-ANAME.

* GET A NEW CUSTOMER NUMBER.
     MOVE 1 TO PAGE-NUM OF CUST-AKEY
     MOVE 1 TO RECORD-NUM OF CUST-AKEY.
     FETCH CUSTNUMS RECORD
         ON ERROR GO TO 2299-EXIT.
     ADD 1 TO NEXT-CUSTNUM.
     MODIFY CUSTNUMS RECORD.
     MOVE NEXT-CUSTNUM TO WRK-CUSTNUM.

* MOVE NEW RECORD DATA FROM WORK ARE TO CUST MAST AREA.
     PERFORM 4200-MOVE-TO-REC.
* ESTABLISH CURRENCY ON LOCATOR RECORDS
     PERFORM 3000-GET-LOCATORS.
* STORE THE NEW RECORD.
     DISPLAY 'STORING NEW CUST-ADDR' UPON PRINTER.
     STORE CUST-MAST.
     MOVE 'OK000' TO OB-HEAD.
     MOVE 5 TO WINQ-LEN. DEPART.
     PERFORM 8200-WINQ-PUT.
     GO TO 2299-EXIT.
2299-EXIT.
EXIT.

*****************************************************
* DELETE AN ADDRESS RECORD BY CUSTOMER NUMBER
*****************************************************
 2300-DEL-ADDR SECTION.
 2301-DA.
     IMPART.
     OPEN CUSTAREA USAGE-MODE IS UPDATE.
     OPEN SCLOCAREA USAGE-MODE IS UPDATE.
     OPEN ZIPLOCAREA USAGE-MODE IS UPDATE.
```

```
     MOVE 'CUSTAREA' TO CUST-ANAME.
     MOVE CM-DATA TO CUST-NUM-A.
     MOVE CUST-NUM-N TO CM-CUSTNUM.

* GET RTECORD TO BE DELETED.
     FETCH CUST-MAST RECORD
         ON ERROR GO TO 2380-RA.
* DELETE IT.
     DELETE CUST-MAST RECORD.
     MOVE 'OK000' TO OB-HEAD.
     MOVE 5 TO WINQ-LEN.
     DEPART.
     PERFORM 8200-WINQ-PUT.
     GO TO 2399-EXIT.
 2380-RA.
     IF ERROR-NUM = '0013'
         MOVE 'OK999' TO WINQ-BUFF
         MOVE 5 TO WINQ-LEN DEPART
         PERFORM 8200-WINQ-PUT
     ELSE
         GO TO 9000-GEN-ERROR.
 2399-EXIT.
     EXIT.

******************************************************
* RETRIEVE A SELECTION LIST BY STATE/CITY MATCH
******************************************************
 2400-SEARCH-STATE SECTION.
 2401-SS.
     IMPART.
     OPEN CUSTAREA USAGE-MODE IS RETRIEVAL.
     OPEN SCLOCAREA USAGE-MODE IS RETRIEVAL.
     OPEN SCLOCIDX USAGE-MODE IS RETRIEVAL.
     MOVE CM-DATA TO SEARCH-STATE-CITY.
     MOVE SEARCH-STATE TO SC-STATE.
     MOVE SEARCH-CITY TO SC-CITY.
     FETCH ST-CITY-LOC RECORD
         ON ERROR GO TO 2480-SS.
     FETCH FIRST RECORD WITHIN SC-SET SET
         ON ERROR GO TO 2480-SS.
     MOVE SPACES TO OB-BLOCK.
     MOVE 1 TO PTR.
 2410-SS.
* CREATE BLOCKS CONTAINING LIST OF SELECTED RECORDS.
     PERFORM 4300-MOVE-LIST.
     STRING LIST-AREA
         DELIMITED BY SIZE INTO OB-BLOCK
         WITH POINTER PTR.
     IF PTR > 2900
         STRING '^' DELIMITED BY SIZE INTO OB-BLOCK
             WITH POINTER PTR
         MOVE 'OK000' TO OB-HEAD
         COMPUTE WINQ-LEN = PTR + 5
         PERFORM 8200-WINQ-PUT
         MOVE SPACES TO OB-BLOCK
         MOVE 1 TO PTR
         PERFORM 8100-WINQ-GET
         IF CM-OP NOT = 'MO'
             GO TO 2490-SS.
     FETCH NEXT RECORD WITHIN SC-SET SET
         AT END GO TO 2450-SS.
     GO TO 2410-SS.
 2450-SS.

* FINISH LAST BLOCK.
     STRING '^' DELIMITED BY SIZE INTO OB-BLOCK
         WITH POINTER PTR.
     MOVE 'OK001' TO OB-HEAD.
```

```
     COMPUTE WINQ-LEN = PTR + 5.
     PERFORM 8200-WINQ-PUT.
     GO TO 2490-SS.
 2480-SS.
     IF ERROR-NUM NOT = '0013'
         GO TO 9000-GEN-ERROR.
     MOVE 'ER999' TO WINQ-BUFF.
     MOVE 5 TO WINQ-LEN.
     PERFORM 8200-WINQ-PUT.
 2490-SS.
     DEPART.
 2499-EXIT.
     EXIT.

******************************************************
* RETRIEVE A SELECTION LIST BY ZIP CODE MATCH
******************************************************
 2500-SEARCH-ZIP SECTION.
 2501-SZ.
     IMPART.
     OPEN CUSTAREA USAGE-MODE IS RETRIEVAL.
     OPEN ZIPLOCAREA USAGE-MODE IS RETRIEVAL.
     OPEN ZIPLOCIDX USAGE-MODE IS RETRIEVAL.
     MOVE CM-DATA TO SEARCH-ZIP.
     MOVE SEARCH-ZIP TO ZL-ZIP.
     FETCH ZIP-LOC RECORD
         ON ERROR GO TO 2580-SZ.
     FETCH FIRST RECORD WITHIN ZIP-SET SET
         ON ERROR GO TO 2580-SZ.
     MOVE SPACES TO OB-BLOCK.
     MOVE 1 TO PTR.
 2510-SZ.
* CREATE BLOCKS CONTAINING LIST OF SELECTED RECORDS.
     PERFORM 4300-MOVE-LIST.
     STRING LIST-AREA
         DELIMITED BY SIZE INTO OB-BLOCK
         WITH POINTER PTR.
     IF PTR > 2900
         STRING '^' DELIMITED BY SIZE INTO OB-BLOCK
             WITH POINTER PTR
         MOVE 'OK000' TO OB-HEAD
         COMPUTE WINQ-LEN = PTR + 5
         PERFORM 8200-WINQ-PUT
         MOVE SPACES TO OB-BLOCK
         MOVE 1 TO PTR PERFORM 8100-WINQ-GET
         IF CM-OP NOT = 'MO'
             GO TO 2590-SZ.
         FETCH NEXT RECORD WITHIN ZIP-SET SET
             AT END GO TO 2550-SZ.
             GO TO 2510-SZ.
 2550-SZ.
* FINISH LAST BLOCK
     STRING '^' DELIMITED BY SIZE INTO OB-BLOCK
         WITH POINTER PTR.
     MOVE 'OK001' TO OB-HEAD.
     COMPUTE WINQ-LEN = PTR + 5.
     PERFORM 8200-WINQ-PUT.
     GO TO 2590-SZ.
 2580-SZ.
     IF ERROR-NUM NOT = '0013'
         GO TO 9000-GEN-ERROR.
     MOVE 'ER999' TO WINQ-BUFF.
     MOVE 5 TO WINQ-LEN.
     PERFORM 8200-WINQ-PUT.
 2590-SZ.
     DEPART.
 2599-EXIT.
```

```
     EXIT.

*****************************************************
* RETRIEVE ALL CUST-MAST RECORDS AND WRITE TO A
* SEQUENTIAL FILE FOR DOWNLOAD BY WINQ
*****************************************************
 2600-DOWNLOAD-FILE SECTION. 2601-DL.
     OPEN OUTPUT WORKFILE.
     IMPART.
     OPEN CUSTAREA USAGE-MODE IS RETRIEVAL.
     FETCH FIRST CUST-MAST WITHIN CUSTAREA AREA
         ON ERROR GO TO 2680-DL.
 2610-DL.
     PERFORM 4100-MOVE-TO-WORK.
     WRITE WORKREC FROM WORK-AREA.
     FETCH NEXT CUST-MAST WITHIN CUSTAREA AREA
         AT END GO TO 2680-DL.
     GO TO 2610-DL.
 2680-DL.
     CLOSE WORKFILE.
     IF ERROR-NUM = '0013' OR '0007'
         NEXT SENTENCE
     ELSE
         GO TO 9000-GEN-ERROR.
     DEPART.
     MOVE 'OK000' TO WINQ-BUFF.
     MOVE 5 TO WINQ-LEN.
     PERFORM 8200-WINQ-PUT.
 2699-EXIT.
     EXIT.

*****************************************************
* ESTABLISH CURRENCY ON LOCATOR RECORDS. IT MAY BE
* NECESSARY TO STORE NEW LOCATOR RECORDS ALSO.
*****************************************************
 3000-GET-LOCATORS SECTION.
 3001-GL.
     MOVE CM-STATE TO SC-STATE.
     MOVE CM-CITY TO SC-CITY.
     MOVE CM-ZIP TO ZL-ZIP.
* GET THE STATE-CITY LOCATOR RECORD
     FETCH ST-CITY-LOC RECORD ON ERROR GO TO 3010-GL.
     GO TO 3020-GL.
 3010-GL.
* IF NO STATE-CTIY LOCATOR, ADD ONE.
     IF ERROR-NUM NOT = '0013'
         GO TO 9000-GEN-ERROR.
     STORE ST-CITY-LOC.

 3020-GL.
* GET ZIP LOCATOR RECORD.
     FETCH ZIP-LOC RECORD ON ERROR GO TO 3030-GL.
     GO TO 3099-EXIT.
     3030-GL.
* IF NO ZIP LOCATOR, ADD ONE.
     IF ERROR-NUM NOT = '0013'
     GO TO 9000-GEN-ERROR.
     STORE ZIP-LOC.
 3099-EXIT.
     EXIT.

*****************************************************
* MOVE CUST-MAST DATA TO WORK AREA.
*****************************************************
 4100-MOVE-TO-WORK SECTION.
 4101-MW.
```

```
     MOVE CM-CUSTNUM TO WRK-CUSTNUM.
     MOVE CM-LAST TO WRK-LAST.
     MOVE CM-FIRST TO WRK-FIRST.
     MOVE CM-MI TO WRK-MI.
     MOVE CM-COMPANY TO WRK-COMPANY.
     MOVE CM-ADDR-1 TO WRK-ADDR1.
     MOVE CM-ADDR-2 TO WRK-ADDR2,
     MOVE CM-ADDR-3 TO WRK-ADDR3.
     MOVE CM-CITY TO WRK-CITY.
     MOVE CM-STATE TO WRK-STATE.
     MOVE CM-ZIP TO WRK-ZIP.
     MOVE CM-AREA-CODE TO WRK-ACODE.
     MOVE CM-PH-NUM TO WRK-PHONE.
     MOVE CM-EXT TO WRK-EXT.
 4199-EXIT.
     EXIT.

******************************************************
* MOVE WORK AREA TO CUST-MAST RECORD AREA.
******************************************************
 4200-MOVE-TO-REC SECTION.
 4201-MR.
     MOVE WRK-CUSTNUM TO CM-CUSTNUM.
     MOVE WRK-LAST TO CM-LAST.
     MOVE WRK-FIRST TO CM-FIRST.
     MOVE WRK-MI TO CM-MI.
     MOVE WRK-COMPANY TO CM-COMPANY.
     MOVE WRK-ADDR1 TO CM-ADDR-1.
     MOVE WRK-ADDR2 TO CM-ADDR-2,
     MOVE WRK-ADDR3 TO CM-ADDR-3.
     MOVE WRK-CITY TO CM-CITY.
     MOVE WRK-STATE TO CM-STATE.
     MOVE WRK-ZIP TO CM-ZIP.
     MOVE WRK-ACODE TO CM-AREA-CODE.
     MOVE WRK-PHONE TO CM-PH-NUM.
     MOVE WRK-EXT TO CM-EXT.
 4299-EXIT.
     EXIT.

******************************************************
* MOVE CUST MAST DATA TO LIST AREA
******************************************************
 4300-MOVE-LIST SECTION.
 4301-ML.
     MOVE CM-CUSTNUM TO LA-CUSTNUM.
     MOVE CM-LAST TO LA-LAST.
     MOVE CM-MI TO LA-MI.
     MOVE CM-FIRST TO LA-FIRST.
     MOVE CM-CITY TO LA-CITY.
     MOVE CM-STATE TO LA-STATE.
     MOVE CM-ZIP TO LA-ZIP.
 4399-EXIT.
     EXIT.

******************************************************
*** WINQ API GET AND PUT MESSAGES FROM/TO THE CLIENT.
******************************************************
 8100-WINQ-GET SECTION.
 8100-WG.
*** GET A WINQ INPUT MESSAGE FROM THE PC WINQ APP.
     ENTER MASM 'WINQ_SVRGET' USING WINQ-STAT WINQ-BUFF WINQ-LEN.
     IF WINQ-STAT > 0
         DISPLAY '******* WINQ GET ERROR:' WINQ-STAT
         IF IMPART-DEPART = 1
             DEPART WITH ROLLBACK
             STOP RUN
         ELSE
             STOP RUN.
     MOVE WINQ-BUFF TO CMD-ACTION.
```

```
 8199-EXIT.
     EXIT.

 8200-WINQ-PUT SECTION.
 8200-WP.
*** SEND A WINQ MESSAGE TO THE PC WINQ APP.
     ENTER MASM 'WINQ_SVRPUT' USING WINQ-STAT WINQ-BUFF WINQ-LEN.
     IF WINQ-STAT > 0
         DISPLAY '******* WINQ PUT ERROR:' WINQ-STAT
         IF IMPART-DEPART = 1
             DEPART WITH ROLLBACK
             STOP RUN
         ELSE
             STOP RUN.
 8299-EXIT.
     EXIT.

*****************************************************
* DMS-1100 GENERAL ERROR AND ROLLBACK HANDLING.
*****************************************************
 9000-DMS-ERROR SECTION.
 9000-GEN-ERROR.
     MOVE 'ER999 DMS-1100 ERROR' TO WINQ-BUFF.
     MOVE 30 TO WINQ-LEN.
     PERFORM 8200-WINQ-PUT.
     PERFORM 9200-DMS-DISPLAY.
     DEPART WITH ROLLBACK.
     STOP RUN.
 9100-ROLL-BACK.
     MOVE 'ER999 DMS-1100 ROLL BACK ERROR' TO WINQ-BUFF.
     MOVE 30 TO WINQ-LEN.
     PERFORM 8200-WINQ-PUT.
     PERFORM 9200-DMS-DISPLAY.
     STOP RUN.
 9200-DMS-DISPLAY.
     ENTER MASM 'WINQ_SVRTERM'.
     DISPLAY 'ERROR-STATUS:' ERROR-STATUS UPON PRINTER.
     DISPLAY 'ERROR-NUM:' ERROR-NUM UPON PRINTER.
     DISPLAY 'ERROR-AREA:' ERROR-AREA UPON PRINTER.
     DISPLAY 'ERROR-REC:' ERROR-RECORD UPON PRINTER.
     DISPLAY 'ERROR-SET:' ERROR-RECORD UPON PRINTER.

 9999-EXIT.
      EXIT.
*****************************************************
* END OF PROGRAM
*****************************************************
```

9.1.3 The Client Program

The Windows Visual Basic program has three forms (**CUSTMAIN.FRM**, **CUSTSO.FRM** and **CUSTLIST.FRM**) and a **CUSTADDR.BAS** file contain parameters global to all functions and subroutines. The initial form displayed is the user sign-on form, **CUSTSO.FRM** (see the forms beginning on the next page). In addition, the code runs the **WinQDem.bas** script to connect and sign on to the host and execute the COBOL program.

The following code is listed by file:

9.1.3.1 CUSTADDR.BAS

```
Option Explicit
' Full address record
Type ADDR_REC_TYPE
    CustNum As String * 10
    Last As String * 20
    First As String * 20
```

```
    MI As String * 1
    Company As String * 30
    Addr1 As String * 30
    Addr2 As String * 30
    Addr3 As String * 30
    City As String * 20
    St As String * 2
    Zip As String * 10
    ACode As String * 3
    Phone As String * 7
    Ext As String * 5
End Type

' Partial address record for selection list
Type ADDR_LIST_TYPE
    CustNum As String * 10
    Last As String * 20
    First As String * 20
    MI As String * 1
    City As String * 20
    St As String * 2
    Zip As String * 10
End Type

Type WINQ_SHORT_TYPE
    Cmd As String * 2 ' Host server command code
    Dta As String * 30 ' Data string for command
End Type

Type WINQ_LONG_TYPE
    Cmd As String * 2 ' Host server command code
    Rec As ADDR_REC_TYPE ' Rec to add or replace
End Type

Type GEN_TYPE
    Rec As String * 300
End Type
```

9.1.3.2 CUSTSO.FRM

```
Option Explicit

Private Sub Btn_browse_Click()
    Dlg_Open.InitDir = " C:\Program Files\KMSystems\UTSPLUS32\4.0\Scripts"
    Dlg_Open.ShowOpen
    If Dlg_Open.filename <> "" Then
        Txt_Script = Dlg_Open.filename
    ElseIf
        Dlg_Open.filename = "" Then
            Txt_Script = "<< No Script Selected >>"
    End If
End Sub

Private Sub Btn_Cancel_Click()
    End
End Sub

Private Sub Btn_OK_Click()
    Dim Rslt As Integer

    ' Verify the User, Pass, Route, and Scripts have been entered.
    If Txt_UserId.Text = "" Then
        MsgBox "A User-Id must be entered to continue.", vbOKOnly + vbInformation, "Missing
Field"
```

```
        Exit Sub
    End If
    If Txt_Password.Text = "" Then
        MsgBox "A Password must be entered to continue.", vbOKOnly + vbInformation, "Missing
Field"
        Exit Sub
    End If
    If Txt_Route.Text = "" Then
        MsgBox "A Route must be entered to continue.", vbOKOnly + vbInformation, "Missing Field"
        Exit Sub
    End If
    If Txt_Script.Text = "<< No Script Selected >>" Then
        MsgBox "A Sign-on Script must be selected to continue.", vbOKOnly + vbInformation,
"Missing Field"
        Exit Sub
    End If

    ' Disable Sign On Controls
    Btn_OK.Enabled = False
    Btn_Cancel.Enabled = False
    CustSO.MousePointer = 11 ' vbHourglass

    ' Open connection to host
    Rslt = CustMain.WinQ32.OpenSession(Txt_Route.Text)
    If Rslt = 0 Then
        MsgBox "WinQ OpenSession Failed", vbExclamation, "WinQ Error"
        CustMain.WinQ32.SSResult = False
        MousePointer = vbDefault
        Exit Sub
    End If

    ' Open and run script that has the user-id and password passed to it

    CustMain.WinQ32.TraceOption = 1  '  Turn trace on during development

    Rslt = CustMain.WinQ32.RunScript(Txt_Script, 2, Trim(Txt_UserId),
Trim(Txt_Password), "", "", "", "")
    If Rslt = 0 Then
        MsgBox "WinQ SignOn Script returned False", vbExclamation, "WinQ
SignOn Error"
        CustMain.WinQ32.SSResult = False
        MousePointer = vbDefault
        Exit Sub
    End If

    ' Sign On was successful, enable controls.
    MousePointer = vbDefault
    Btn_OK.Enabled = True
    Btn_Cancel.Enabled = True
    Hide
    CustMain.SSResult = True
End Sub

Private Sub Form_Load()
    Move (Screen.Width -Width) / 2, (Screen.Height -Height) / 2 ' Center
                                                                ' form
End Sub

Private Sub Txt_Password_KeyPress(KeyAscii As Integer)
    KeyAscii = Asc(UCase$(Chr$(KeyAscii)))
End Sub

Private Sub Txt_UserId_KeyPress(KeyAscii As Integer)
    KeyAscii = Asc(UCase$(Chr$(KeyAscii)))
End Sub
```

9.1.3.3 CUSTMAIN.FRM

```
Option Explicit

Public SSResult As Boolean
Dim Retrieved As Integer

Public Cancelled As Integer
Public Signed_On As Integer
Public Script_Dir As String
Public Selected_Cust_Ent As Integer
Public Sel_List_Cnt As Integer
Dim Sel_List(1 To 200) As ADDR_LIST_TYPE


Private Sub Btn_Add_Click()
    Dim AddCmd As WINQ_LONG_TYPE
    Dim Msg As GEN_TYPE
    Dim Rep As String * 30
    Dim Rslt As Integer
    ' Confirm before adding new record
    If MsgBox("Are you sure you want to add this as a new record?", vbYesNo, "Confirm") = vbNo
Then
        Exit Sub
    End If
    ' Build the WinQ command message to do an ADD.
    AddCmd.Cmd = "AD"
    AddCmd.Rec.CustNum = "0000000000"
    AddCmd.Rec.Last = Txt_Last
    AddCmd.Rec.First = Txt_First
    AddCmd.Rec.MI = Txt_MI
    AddCmd.Rec.Company = Txt_Comp
    AddCmd.Rec.Addr1 = Txt_Addr1
    AddCmd.Rec.Addr2 = Txt_Addr2
    AddCmd.Rec.Addr3 = Txt_Addr3
    AddCmd.Rec.City = Txt_City
    AddCmd.Rec.St = Txt_St
    AddCmd.Rec.Zip = Txt_Zip
    AddCmd.Rec.ACode = Txt_Acode
    AddCmd.Rec.Phone = Txt_Phone
    AddCmd.Rec.Ext = Txt_Ext

    LSet Msg = AddCmd
    ' Send the WinQ Command and get the host response.
    If Not Host_Command(Msg.Rec, Rep, 30) Then
        Exit Sub
    End If
    If Left$(Rep, 5) = "OK000" Then
        Txt_CustNum = Mid$(Rep, 6, 10)
        MsgBox "Record Added.", vbInformation
        Retrieved = False
    Else
        MsgBox "Record not added:" + Trim$(Rep), vbExclamation
    End If
End Sub


Private Sub Btn_Clear_Click()
    ' Clear contontents of all entry boxes.
    Txt_CustNum = ""
    Txt_First = ""
    Txt_MI = ""
    Txt_Last = ""
    Txt_Comp = ""
    Txt_Addr1 = ""
    Txt_Addr2 = ""
    Txt_Addr3 = ""
```

```
    Txt_City = ""
    Txt_St = ""
    Txt_Zip = ""
    Txt_Acode = ""
    Txt_Phone = ""
    Txt_Ext = ""
    Retrieved = False
End Sub


Private Sub Btn_Close_Click()
    If MsgBox("Are you sure you want to close now?", vbQuestion + vbYesNo) = vbYes Then Hide
    End If
End Sub


Private Sub Btn_Delete_Click()
    Dim Rep As String * 50

    If Not Host_Command("DT" + Format$(Val(Txt_CustNum), "0000000000"), Rep, 30) Then
        Exit Sub
    End If
    If Left$(Rep, 5) <> "OK000" Then
        MsgBox "Record not Deleted." + Trim$(Rep), vbInformation
        Exit Sub
    End If
    MsgBox "Record Deleted.", vbInformation

    Retrieved = True
    Btn_Clear_Click
End Sub


Private Sub Btn_Download_Click()
    Dim PC_File As String
    Dim tmp As String
    Dim Rslt As Integer

    ' Use common dialog to select download file.
    Cmdialog1.DialogTitle = "Select Downlod file"
    Cmdialog1.Filter = "Text files(*,txt)|*.TXT|All files(*.*)|*.*"
    Cmdialog1.FilterIndex = 1
    Cmdialog1.Flags = cdlOFNHideReadOnly Or cdlOFNPathMustExist
    Cmdialog1.CancelError = True
    On Error Resume Next
    Cmdialog1.Action = 1
    '    If Err = CDERR_CANCEL Then
    '        Exit Sub
    '    ElseIf Err <> 0 Then
    '        MsgBox "Dialog error:" + Error$
    '        Exit Sub
    '    End If
    PC_File = Cmdialog1.filename

    ' Tell host server program to create a download work file.
    If Not Host_Command("DL", tmp, 120) Then
        Exit Sub
    End If
    If tmp <> "OK000" Then
        MsgBox "Download work file could not be created on host.", vbExclamation
    End If

    ' Download the file created by the host server.
    Rslt = WinQ32.GetFile("WORKFILE", PC_File, 1, 300)
    If Rslt = 0 Then
        MsgBox "Download failed", "", vbExclamation
    Else
```

```
        MsgBox "Address download completed", vbInformation
    End If
End Sub


Private Sub Btn_Replace_Click()
    Dim RepCmd As WINQ_LONG_TYPE
    Dim Msg As GEN_TYPE
    Dim Rep As String * 30
    Dim Rslt As Integer
    RepCmd.Cmd = "RP"
    RepCmd.Rec.CustNum = Format$(Val(Txt_CustNum), "0000000000")
    RepCmd.Rec.Last = Txt_Last
    RepCmd.Rec.First = Txt_First
    RepCmd.Rec.MI = Txt_MI
    RepCmd.Rec.Company = Txt_Comp
    RepCmd.Rec.Addr1 = Txt_Addr1
    RepCmd.Rec.Addr2 = Txt_Addr2
    RepCmd.Rec.Addr3 = Txt_Addr3
    RepCmd.Rec.City = Txt_City
    RepCmd.Rec.St = Txt_St
    RepCmd.Rec.Zip = Txt_Zip
    RepCmd.Rec.ACode = Txt_Acode
    RepCmd.Rec.Phone = Txt_Phone
    RepCmd.Rec.Ext = Txt_Ext

    LSet Msg = RepCmd

    If Not Host_Command(Msg.Rec, Rep, 30) Then
        Exit Sub
    End If

    If Left$(Rep, 5) = "OK000" Then
        Txt_CustNum = Mid$(Rep, 6, 10)
        MsgBox "Record Replaced.", vbInformation
        Retrieved = False
    Else
        MsgBox "Record not replaced:" + Trim$(Rep), vbExclamation
    End If
End Sub


Private Sub Btn_Retrieve_Click()
    Dim RetRec As ADDR_REC_TYPE
    Dim Rep As GEN_TYPE
    Dim Rslt As Integer

    If Not Host_Command("RT" + Format$(Val(Txt_CustNum), "0000000000"), Rep.Rec, 30) Then
        Exit Sub
    End If
    If Left$(Rep.Rec, 5) <> "OK000" Then
        MsgBox "Record not found." + Trim$(Rep.Rec), vbInformation
        Exit Sub
    End If
    Rep.Rec = Mid$(Rep.Rec, 6, Len(Rep.Rec) -6)
    LSet RetRec = Rep

    Txt_CustNum = RTrim$(RetRec.CustNum)
    Txt_Last = RTrim$(RetRec.Last)
    Txt_First = RTrim$(RetRec.First)
    Txt_MI = RTrim$(RetRec.MI)
    Txt_Comp = RTrim$(RetRec.Company)
    Txt_Addr1 = RTrim$(RetRec.Addr1)
    Txt_Addr2 = RTrim$(RetRec.Addr2)
    Txt_Addr3 = RTrim$(RetRec.Addr3)
    Txt_City = RTrim$(RetRec.City)
    Txt_St = RTrim$(RetRec.St)
    Txt_Zip = RTrim$(RetRec.Zip)
```

```
    Txt_Acode = RTrim$(RetRec.ACode)
    Txt_Phone = RTrim$(RetRec.Phone)
    Txt_Ext = RTrim$(RetRec.Ext)

    Beep

    Retrieved = True
End Sub


Private Sub Btn_Search_SC_Click()
    If (Txt_St = "") Or (Txt_City = "") Then
        MsgBox "City and State fields must be entered to search
records.", vbInformation, "Search Parameters Not Entered"
        Exit Sub
    End If
    Cust_Search "SS", Trim$(Txt_St), Trim$(Txt_City)
    If Selected_Cust_Ent > 0 Then
        Txt_CustNum = Sel_List(Selected_Cust_Ent).CustNum

        Btn_Retrieve_Click
    End If \
End Sub


Private Sub Btn_Search_Zip_Click()
    If Txt_Zip = "" Then
        MsgBox "Zip field must be entered to search records.",
        vbInformation, "Search Parameters Not Entered"
        Exit Sub
    End If
    Cust_Search "SZ", Trim$(Txt_Zip), ""
    If Selected_Cust_Ent > 0 Then
        Txt_CustNum = Sel_List(Selected_Cust_Ent).CustNum
        Btn_Retrieve_Click
    End If
End Sub


Private Sub Form_Load()
    Move (Screen.Width -Width) / 2, (Screen.Height -Height) / 2 ' Center
                                                                ' form
    ' Set some initial WinQ32 modes.
    WinQ32.ScreenVisible = 1  ' Make screen visible during development
    WinQ32.OperationMode = 1

    ' Show the Sign On Window.
    CustSO.Show 1
    If SSResult <> True Then
        Exit Sub
    End If

    ' Show the Main Activity Window.
    Show 1
    If SSResult <> True Then
        Exit Sub
    End If

   ' End Application
    WinQ32.CloseSession
    End

End Sub


Private Sub Form_Unload(Cancel As Integer)
```

```
    Cancel = True
    Btn_Close_Click
End Sub


Private Sub Txt_ACode_KeyPress(KeyAscii As Integer)
    Dim Ch As String * 1

    If KeyAscii = vbKeyBack Then Exit Sub
    Ch = Chr$(KeyAscii)
    If Ch < "0" Or Ch > "9" Then
        Beep
        KeyAscii = 0
    End If
End Sub


Private Sub Txt_City_KeyPress(KeyAscii As Integer)
    KeyAscii = Asc(UCase$(Chr$(KeyAscii)))
End Sub


Private Sub Txt_CustNum_KeyPress(KeyAscii As Integer)
    Dim Ch As String * 1

    If Retrieved Then
        MsgBox "Customer number can not be changed. Use ""Clear Entries""
button to clear all entries before entering a new customer number.",
vbExclamation
        KeyAscii = 0
        Exit Sub
    End If

    If KeyAscii = vbKeyBack Then Exit Sub
    Ch = Chr$(KeyAscii)
    If Ch < "0" Or Ch > "9" Then
        Beep
        KeyAscii = 0
    End If
End Sub


Private Sub Txt_Ext_KeyPress(KeyAscii As Integer)
    Dim Ch As String * 1
    If KeyAscii = vbKeyBack Then Exit Sub
    Ch = Chr$(KeyAscii)
    If Ch < "0" Or Ch > "9" Then
        Beep
        KeyAscii = 0
    End If
End Sub


Private Sub Txt_Last_KeyPress(KeyAscii As Integer)
    KeyAscii = Asc(UCase$(Chr$(KeyAscii)))
End Sub


Private Sub Txt_Phone_KeyPress(KeyAscii As Integer)
    Dim Ch As String * 1
    If KeyAscii = vbKeyBack Then Exit Sub
    Ch = Chr$(KeyAscii)
    If Ch < "0" Or Ch > "9" Then
        Beep
        KeyAscii = 0
```

```
    End If
End Sub


Private Sub Txt_St_KeyPress(KeyAscii As Integer)
    KeyAscii = Asc(UCase$(Chr$(KeyAscii)))
End Sub


Private Sub Txt_Zip_KeyPress(KeyAscii As Integer)
    Dim Ch As String * 1

    If KeyAscii = vbKeyBack Then Exit Sub
    Ch = Chr$(KeyAscii)
    If Ch < "0" Or Ch > "9" And Ch <> "-" Then
        Beep
        KeyAscii = 0
    End If
End Sub


Sub Cust_Search(Func As String, P1 As String, P2 As String)
    ' Send command to host to do search and format blocks of
    ' records to be listed for selection.
    Dim l As Integer
    Dim X As Integer
    Dim Rslt As Integer
    Dim Done As Integer
    Dim First As Integer
    Dim Reply As String
    Dim Msg As String
    Dim Count As Integer
    Dim Ent As GEN_TYPE

    First = True
    Sel_List_Cnt = 0
    ' Format fisrt host message to start retrieval
    ' by State and City or Zip Code.
    Msg=Func+P1+P2
    Do
        If Not First Then Msg = "MO" ' Send more
        ' Send command to host
        If Not Host_Command(Msg, Reply, 60) Then
            Selected_Cust_Ent = 0
            Exit Sub
        End If
        If Left$(Reply, 2) <> "OK" Then
            Selected_Cust_Ent = 0
            Exit Sub
        End If
        First = False

        If Left$(Reply, 5) = "OK001" Then ' Last block, else
                                          ' not last block.
            Done = True
        Else
            Done = False
        End If

        X = InStr(Reply, "^")          ' Find end of block indicator.
        If X = 0 Then X = Len(Reply)   ' No EOB try to use what's there
        Reply = Mid$(Reply, 6,X-6) ' Drop hdr and EOB ind.
```

```
        ' Break block into individual records.
        l = Len(Sel_List(1)) ' Lenght of one entry
        X=1
        While X < Len(Reply)
            Sel_List_Cnt = Sel_List_Cnt + 1
            Ent.Rec = Mid$(Reply, X, l)
            LSet Sel_List(Sel_List_Cnt) = Ent
            X = X + l
        Wend
    Loop Until Done

    ' All selected customer addresses retrieved.
    Sort_Selected_List
    ' Add all selected items to list box
    CustList!Lst_Cust.Clear
    For X = 1 To Sel_List_Cnt
        CustList!Lst_Cust.AddItem Trim$(Sel_List(X).Last) + ", " +
Trim$(Sel_List(X).First)+""+ Sel_List(X).MI + ". " +
Trim$(Sel_List(X).City) + ", " + Sel_List(X).St+""+ Sel_List(X).Zip
    Next X
    ' Show selecte list window.
    CustList.Show 1

End Sub


Public Sub Sort_Selected_List()
    ' This subroutine sorts the selected customer list
    ' by last name.

    Dim X As Integer
    Dim Y As Integer
    Dim Hld As ADDR_LIST_TYPE

    If Sel_List_Cnt < 2 Then Exit Sub ' Nothing to sort
    For X = 1 To Sel_List_Cnt -1
        For Y = X + 1 To Sel_List_Cnt
            If Sel_List(Y).Last < Sel_List(X).Last Then
                LSet Hld = Sel_List(X)
                LSet Sel_List(X) = Sel_List(Y)
                LSet Sel_List(Y) = Hld
            End If
        Next Y
    Next X
End Sub


Function Host_Command(Msg As String, Reply As String, Time_Out As Integer) As Integer
    ' This is a general function for sending a single command
    ' to the host server program
    ' and recieving one response from the host server.
    Dim Rslt As Integer
    Dim WinQStatus As String

    Host_Command = False
    Time_Out = 1000

    Rslt = CustMain.WinQ32.SendApp(Msg, Time_Out)
    If Rslt <> 1 Then
        If CustMain.WinQ32.LastStatus = 1 Then
            MsgBox "Time out on SendApp", vbExclamation
        Else
            WinQStatus = "Host connection error: " & CustMain.WinQ32.LastStatus
            MsgBox WinQStatus, vbExclamation
        End If
        Exit Function
```

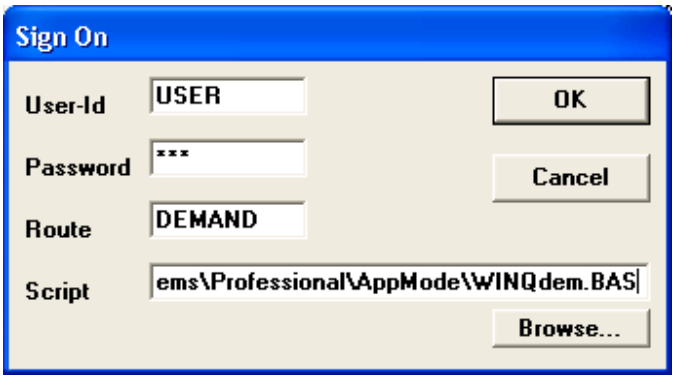
```
    End If
    Reply = CustMain.WinQ32.GetApp(Time_Out)
    If CustMain.WinQ32.LastStatus <> 0 Then
        If CustMain.WinQ32.LastStatus = 1 Then
            MsgBox "Time out on GetApp", vbExclamation
        Else
            WinQStatus = "Host connection error-->: " & CustMain.WinQ32.LastStatus
            MsgBox WinQStatus, vbExclamation
        End If
        Exit Function

    End If
    Host_Command = True
End Function
```

9.1.3.4 CUSTLIST.FRM

```
Option Explicit

Private Sub Btn_Cancel_Click()
    CustMain.Cancelled = True Hide
End Sub


Private Sub Btn_OK_Click()
    If Lst_Cust.ListIndex = -1 Then
        MsgBox "No customer entry selected.", vbExclamation
        Exit Sub
    End If
    CustMain.Cancelled = False
    CustMain.Selected_Cust_Ent = Lst_Cust.ListIndex + 1
    Hide
End Sub


Private Sub Form_Load()
    Move (Screen.Width -Width) / 2, (Screen.Height -Height) / 2 ' Center
                                                                ' form
End Sub


Private Sub Label1_Click()
End Sub


Private Sub Lst_Cust_DblClick()
    Btn_OK_Click
End Sub
```

9.1.3.5 The Start-Up or Sign-On Window

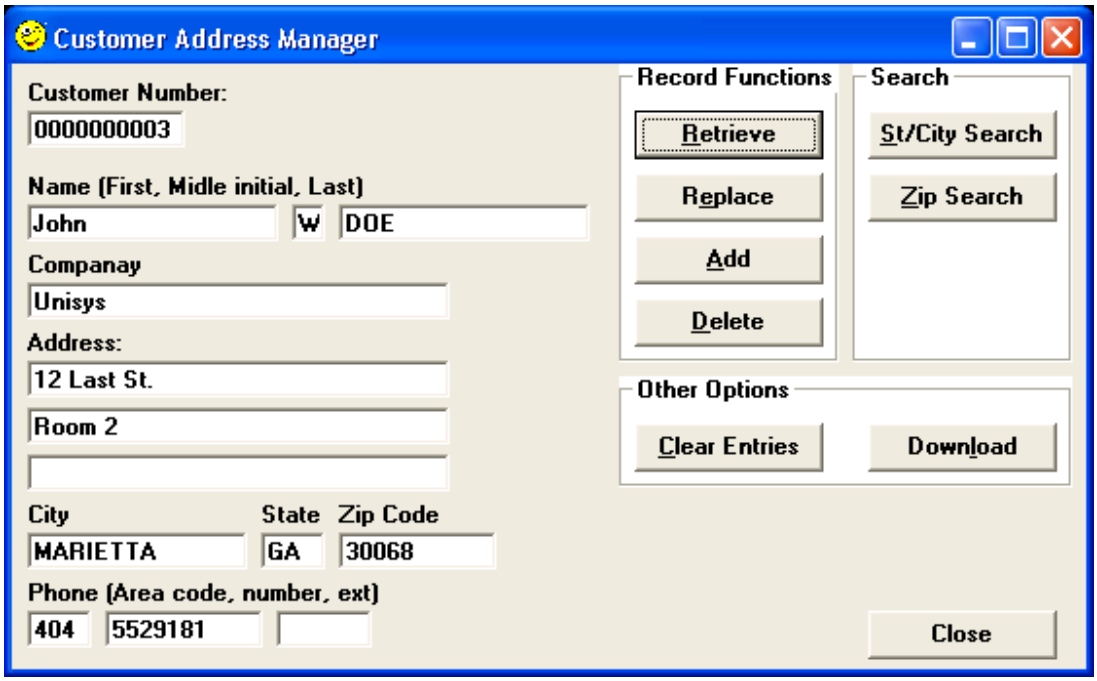
The sign-on window, named **CustSO**, prompts the user for identification. Once the sign on is successful, the main window is displayed.



When the OK button is clicked, the **OpenSession** and **RunScript** functions in **CustSO** are executed. The **RunScript** function runs the named script, which signs on and executes the host program. If the sign-on is successful, the sign-on window is hidden and the **CustMain** window is displayed.

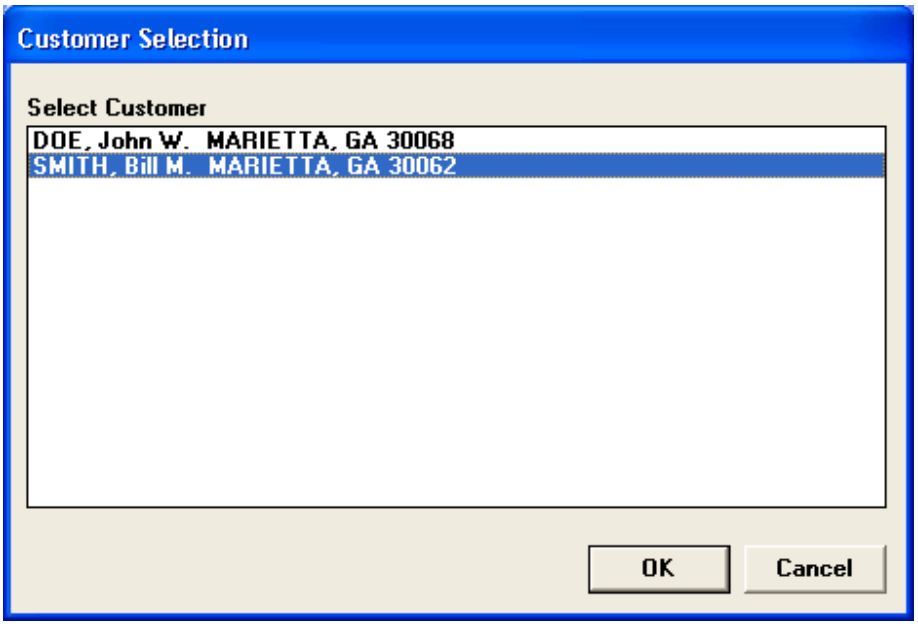
9.1.3.6 The Main Activity Window

The main window provides access to all other functions of this application.



9.1.3.7 The Search Results Window

This window appears as the result of the user pressing the **St/City Search** or **Zip Search** buttons.



9.1.3.8 The Sign-on Script

The following script is run by the **RunScript** function:

```
' *** Script recorded by UTS eXpress Plus Script Recorder
Sub Main()
    dim UserId as String
    dim Password as String
    UserId = GetUserParam(1)
    Password = GetUserParam(2)
    UTSKey UK_TRANSMIT_KEY
    Wait 500
    EnterText Chr$(30) & UserId & "/" & Password
    UTSKey UK_TRANSMIT_KEY
    If not WaitForSpecificString(23, 1, 0, Chr$(30) & "Enter your proj") Then
        Exit Sub
    End If
    UTSKey UK_TRANSMIT_KEY
    If not WaitForSpecificString(23, 1, Chr$(30) & " ", 0) Then
        Exit Sub
    End If
    UTSKey UK_CURSOR_TO_HOME
    UTSKey UK_Erase_DISPLAY
    Wait 1000
    EnterText "@xqt q*pf.winqcob "   `  Your program here.
    UTSKey UK_TRANSMIT_KEY

    Wait 1000
    SetSignOnResult 1
End Sub
```

Note: The script may need to be altered if other prompts are required during the sign-on process.

Chapter 10:CUSTSCHEMA Schema Listing

```
IDENTIFICATION DIVISION
SCHEMA NAME IS CUSTSCHEMA FILE SCHEMA-ABS

DATA DIVISION
DATA NAME SECTION
01 CUST-AKEY USAGE IS AREA-KEY
01 CUST-ANAME USAGE IS AREA-NAME

AREA SECTION
AREA CONTROL IS 600 AREAS
AREA LOOKS INCLUDE AFTER-LOOKS, QUICK-BEFORE-LOOKS

AREA NAME IS CUSTAREA CODE IS 1
    MODE IS DATA AREA
    ALLOCATE 50 PAGES
        EXPANDABLE TO 99999 PAGES
    PAGES ARE 896 WORDS

AREA NAME IS SCLOCAREA CODE IS 2
    MODE IS DATA AREA
    ALLOCATE 56 PAGES 5 OVERFLOW PAGES AT END
        EXPANDABLE TO 99999 PAGES
    PAGES ARE 896 WORDS
    LOAD IS 11 WORDS

AREA NAME IS SCLOCIDX CODE IS 3
    MODE IS INDEX AREA
    ALLOCATE 5 PAGES
        EXPANDABLE TO 99999 PAGES
    PAGES ARE 896 WORDS

AREA NAME IS ZIPLOCAREA CODE IS 4
    MODE IS DATA AREA
    ALLOCATE 55 PAGES 5 OVERFLOW PAGES AT END
        EXPANDABLE TO 99999 PAGES
    PAGES ARE 896 WORDS
    LOAD IS 7 WORDS

AREA NAME IS ZIPLOCIDX CODE IS 5
    MODE IS INDEX AREA
    ALLOCATE 5 PAGES
        EXPANDABLE TO 99999 PAGES
    PAGES ARE 896 WORDS


RECORD SECTION

RECORD NAME IS CUSTNUMS CODE IS 1
    LOCATION MODE IS DIRECT CUST-AKEY, CUST-ANAME
    WITHIN CUSTAREA
```

```
    RECORD MODE IS ASCII COBOL

    05 NEXT-CUSTNUM           PIC 9(10) USAGE COMP
    05 FILLER                 PIC X(12)

RECORD NAME IS CUST-MAST CODE IS 2
    LOCATION MODE IS CALC DMSCALC
        IN CUST-ANAME
        USING CM-CUSTNUM DUPLICATES ARE NOT ALLOWED
    WITHIN CUSTAREA
    RECORD MODE IS ASCII COBOL

    05 CM-CUSTNUM             PIC 9(10) USAGE COMP
    05 CM-NAME
        10 CM-LAST            PIC X(20)
        10 CM-FIRST           PIC X(20)
        10 CM-MI              PIC X
    05 CM-COMPANY             PIC X(30)
    05 CM-ADDRESS
        10 CM-ADDR-1          PIC X(30)
        10 CM-ADDR-2          PIC X(30)
        10 CM-ADDR-3          PIC X(30)
    05 CM-CITY                PIC X(20)
    05 CM-STATE               PIC X(2)
    05 CM-ZIP                 PIC X(10)
    05 CM-PHONE
        10 CM-AREA-CODE       PIC X(3)
        10 CM-PH-NUM          PIC X(7)
        10 CM-EXT             PIC X(5)

RECORD NAME IS ST-CITY-LOC CODE IS 3
    LOCATION MODE IS INDEX SEQUENTIAL
        USING ASCENDING KEY SC-STATE SC-CITY
        INDEX AREA IS SCLOCIDX
        LINKS ARE NEXT
        DUPLICATES ARE NOT ALLOWED
    WITHIN SCLOCAREA
    RECORD MODE IS ASCII COBOL

    05  SC-ST-CITY
        10 SC-STATE PIC X(2)
        10 SC-CITY PIC X(20)
    05  FILLER PIC XX

RECORD NAME IS ZIP-LOC CODE IS 4
    LOCATION MODE IS INDEX SEQUENTIAL
        USING ASCENDING KEY ZL-ZIP
        INDEX AREA IS ZIPLOCIDX
        LINKS ARE NEXT
        DUPLICATES ARE NOT ALLOWED
    WITHIN ZIPLOCAREA
    RECORD MODE IS ASCII COBOL

    05 ZL-ZIP PIC X(5)
    05 FILLER PIC XX

SET SECTION

SET NAME IS SC-SET CODE IS 1
    MODE IS CHAIN LINKED PRIOR
    ORDER IS NEXT
    OWNER IS ST-CITY-LOC
        MEMBER IS CUST-MAST AUTOMATIC
    SELECTION CURRENT SET
```

```
SET NAME IS ZIP-SET CODE IS 2
    MODE IS CHAIN
    LINKED PRIOR
    ORDER IS NEXT
    OWNER IS ZIP-LOC
        MEMBER IS CUST-MAST AUTOMATIC
    SELECTION CURRENT SET
```

Index

If you would like to help us make our documentation better, please take a few moments to complete this form and return it to KMSYS Worldwide. We are always looking for ways to improve our products and your feedback will help us reach our goal.

Name _____

Company _____

Address _____

_____

City _____ State/Province _____

Country _____ Zip/Mail Code _____

Document Name _____ OS Level _____

KMSYS Worldwide Product _____ Level _____

Please rate the documentation on a scale of 1 to 5:

	5	4	3	2	1	
Complete	❑	❑	❑	❑	❑	Incomplete
Accurate	❑	❑	❑	❑	❑	Inaccurate
Usable	❑	❑	❑	❑	❑	Unusable
Readable	❑	❑	❑	❑	❑	Unreadable
Understandable	❑	❑	❑	❑	❑	Unintelligible
Attractive	❑	❑	❑	❑	❑	Unattractive
Excellent	❑	❑	❑	❑	❑	Poor

What information did you expect to find that was omitted?

_____

Is more information needed? ❑ Yes ❑ No. If yes, on what topic?

_____

Did you find factual errors in the documentation? ❑ Yes ❑ No. If yes, please give page number and description of the error.

_____

If the documentation is difficult to understand, please specify page number and problem.

_____

Is the documentation intimidating? ❑ Yes ❑ No.

_____

Are the manuals: ❑ Too long? ❑ Too short? ❑ About the right length?

_____

Other suggestions or comments? (Use back of form if necessary.)

_____

(Additional Comments)

. . . . . Fold along dotted line. . . . . .

KMSYS
WORLDWIDE, INC
P.O. Box 669695
Marietta, GA 30066
U.S.A.

Attn: Technical Documentation Section