UNISYS

Designer Workbench Workshop
Student Guide

The names, places, and/or events used in this publication are purely fictitious and are not intended to correspond to any real individual, group, company, or event. Any similarity or likeness to any real individual, company, or event is purely coincidental and unintentional.

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed license or agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such license or agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded to Unisys Corporation, Education Publications, P.O. Box 1110, Princeton, NJ 08543 U.S.A.

Objectives

Upon completion of this module, you should be able to

1. Define the Designer Workbench product
2. List the features of Designer Workbench
3. Differentiate between the benefits of Designer Workbench as they relate to MAPPER, LINC, and ALLY
4. Describe the benefits of using Designer Workbench
5. List the hardware and software requirements for a Designer Workbench station

Learning Sequence

The goal of this module is to provide a general product overview of Designer Workbench. Even though students should have a grasp of this information (see prerequisites), it can serve as a quick review. The topics for this module include:

- Features
- Documentation
- MAPPER 4R2 / Level 36R1
- LINC II
- ALLY 3.3
- Migration and Co-existence
- Customer Benefits
- End User Benefits
- I. S. Department Benefits
- Workstation Requirements

References

Documentation referenced in this module:

Designer Workbench Capabilities Overview (7831 9746-000)

Learning Sequence

- Features
- Documentation
- MAPPER 4R2 / Level 36R1
- LINC II
- ALLY 3.3
- Migration and Co-existence
- Customer Benefits
- End User Benefits
- I. S. Department Benefits
- Workstation Requirements

Designer Workbench

- **PC client of the 4GL product lines**
  - MAPPER
  - LINC
  - ALLY
- **Delivers the following benefits**
  - Graphical User Interface (GUI)
  - Ease of operation
  - Consistent view across 4GL, mainframe, and PC environments

Designer Workbench Features

- **Graphical User Interface under Microsoft Windows 3.0**
- **Communications to supported Unisys architectures**
  - Uniscope
  - Poll Select
  - TTY
  - TCP/IP
- **Object-oriented repository for CASE and 4GL products**
- **Administration facilities**
- **Full mouse capability with 4GL products**
- **High level interactive Forms Designer**
  - Input/Output forms for MAPPER applications
  - Automatic interface upgrade for LINC systems
  - ALLY within a Windows environment

Designer Workbench Documentation

- **Extensive online documentation**
  - Windows style of HELP
- **Hard copy documentation**
  - Designer Workbench Capabilities Overview
  - Designer Workbench Installation and Administration Guide
  - Designer Workbench Developer Training Guide
  - Designer Workbench Operations Guide
  - INFOConnect UTS 60 Emulator Operating Guide
  - INFOConnect MT Emulator Operating Guide
  - INFOConnect ANSI X3.64 Emulator Operating Guide
- **Demonstration Diskettes**
  - Introduction to Designer Workbench
  - Unisys Designer Workbench CBT

MAPPER 4R2/36R1

- **Forms can be designed, built, and modified with the DW Forms Designer**

- **Non-DW applications can use terminal emulators in Windows**

- **Features include**
  - Pictures/Graphics/Images
  - Button styles
  - Value listings
  - Color

LINC II

- **Workstation Driver Program (WDP) that builds DW screens from LINC screen definitions**

- **Forms can be further customized with the Forms Designer**

- **Existing LINC systems are automatically updated**

- **Features include**
  - Images
  - Color
  - Multiple Ispecs
  - Stylistic text

ALLY 3.3

- **Runs within the Windows environment using its proprietary high level user interface**

- **DW provides full mouse capability**

Migration and Co-existence

- **Migration options**
  - A department at a time
  - Application by application
- **DW and native terminal co-existence**
  - DW MAPPER applications may require minor modifications
  - LINC does not require modifications in REL.1

Designer Workbench Customer Benefits

- **Powerful graphic presentation development tools**

- **Cost reduction associated with delivering end-user applications**
  - Development
  - Training
  - Shortened delivery time

- **Common view to all host and PC based applications**

Designer Workbench End User Benefits

- Consistent, high level view to host and PC applications
- Easy access to host environments
- Reference network information without leaving local workstation applications

I.S. Department Benefits

- **Forms Designer for creation and modification**
  - Delivers complex GUI presentations
  - Includes pictures, graphics, images, LOV, stylistic text
- **Easy to use features reduce development time**
- **Object oriented repository**
  - Workstation client of UREP
  - Foundation for CASE tools integration
  - Reduces network impact

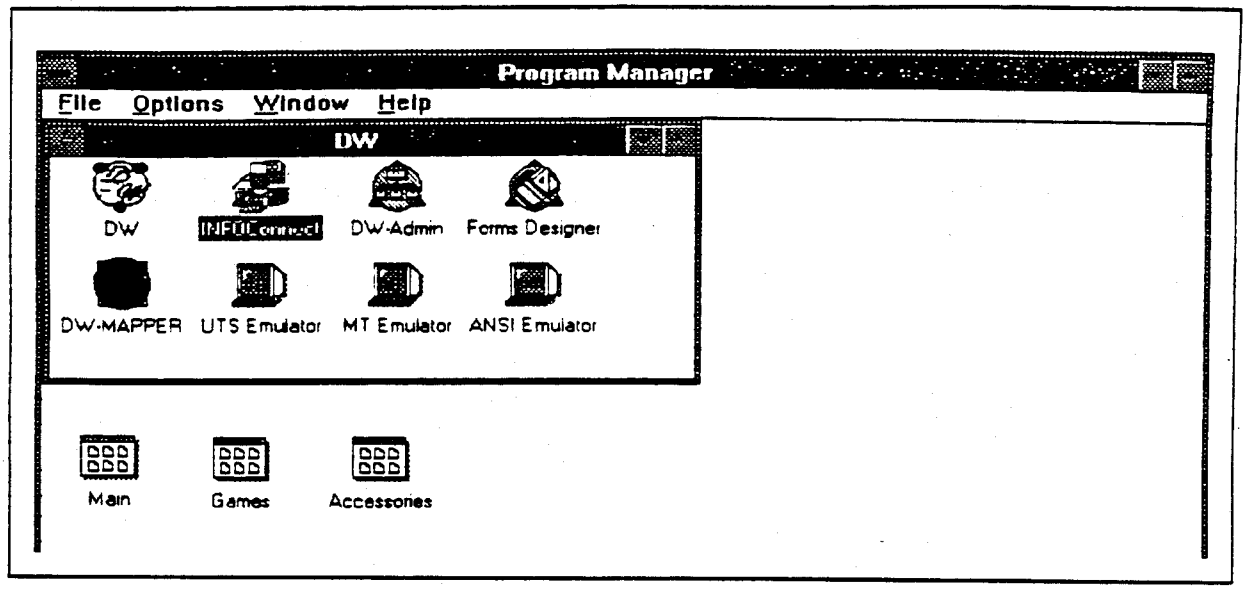
Workstation Requirements

- **Hardware**
  - Unisys 80386 PC (or compatible)
  - EGA or VGA monitor
  - Mouse
  - 4 MB RAM
  - Hard disk with at least 10 MB free space
  - 1.2 MB 5.25-inch or 1.4 MB 3.5-inch diskette drive
  - Appropriate communications hardware for type of connection
    - COM port for TTY or Poll/select
    - STEP board for UNISCOPE
    - ETHERNET board for TCP/IP

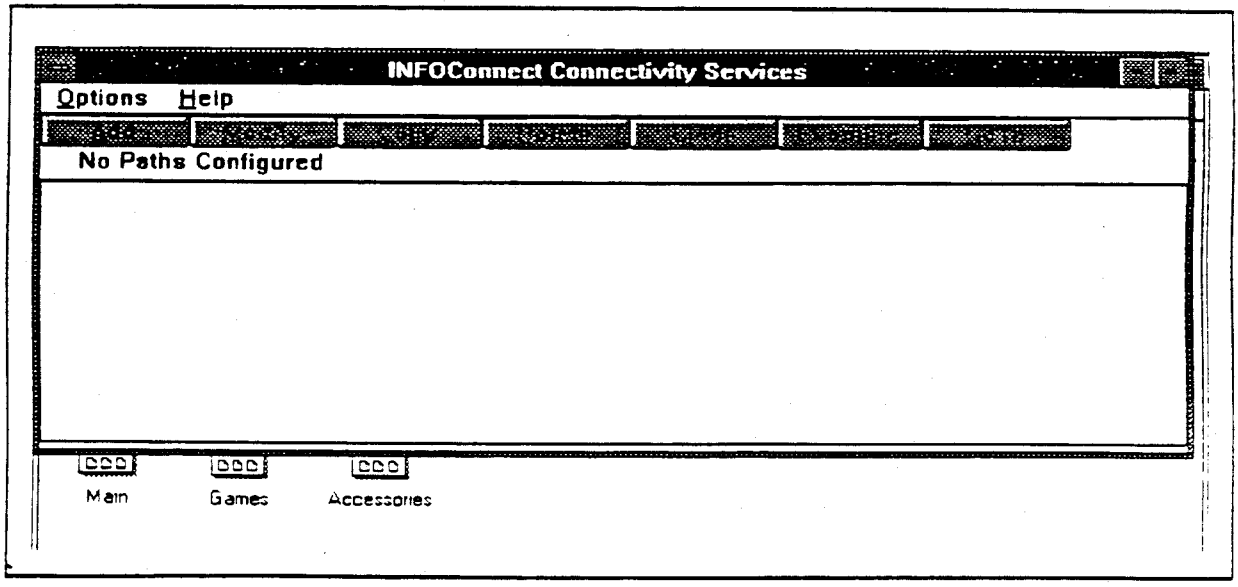
- **Software**
  - PC
    - MS-DOS 3.3 or higher
    - MS-Windows 3.0 or higher
  - Host
    - MAPPER 4R2 or 36R1
    - LINC 14.3 RT with WDP A-Series or LINC 14.4 2200

Creating a Communications Path

- **Select INFOConnect icon from DW window**



- **Select 'Add' from INFOConnect Connectivity Series window**



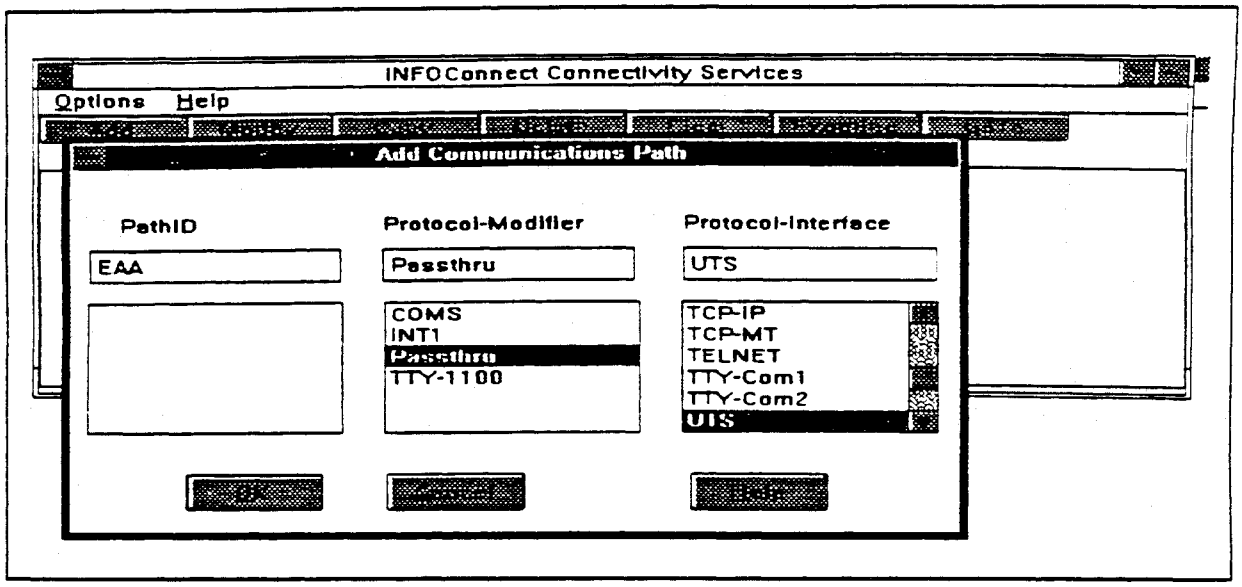
Creating a Communications Path

Visual 1: After requesting to 'Add' a communications path, the Add Communications Path menu is displayed. There are three information items that need to be supplied. In this example, we wish to create a 'Uniscope' protocol path to MAPPER on an 1100 host. We are choosing EAA to be the PathID. We then selected Passthru as the Protocol-Modifier and UTS as the Protocol-Interface. After making the selections, select OK.

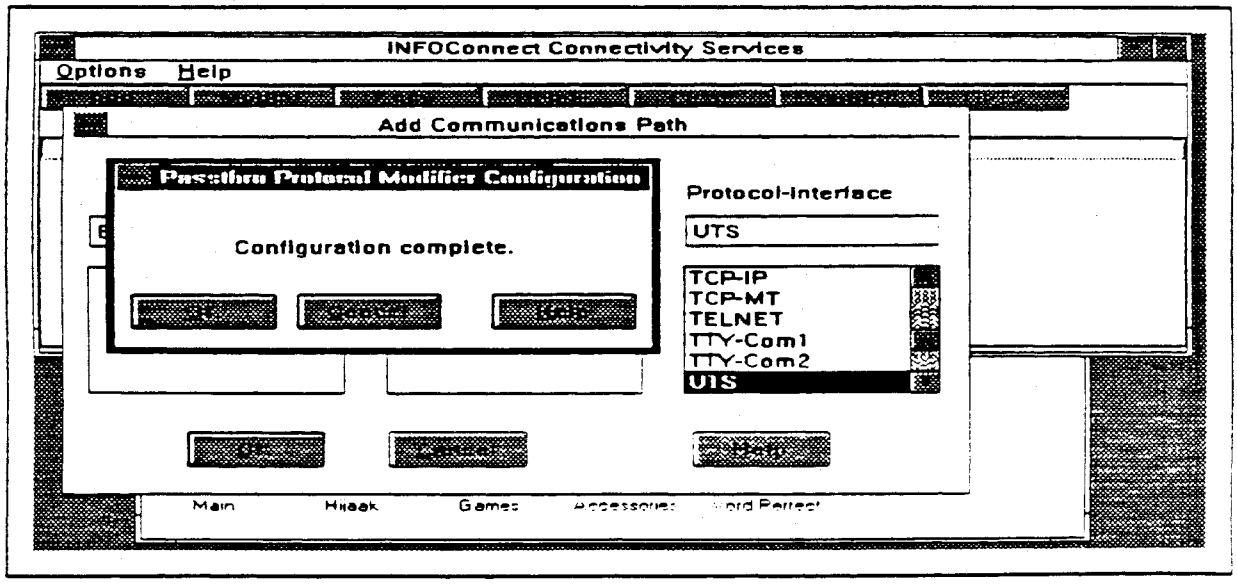
Visual 2: A window will appear to confirm the communication path.

Creating a Communications Path

- Fill in Communication Path window



- Configuration Complete window appears



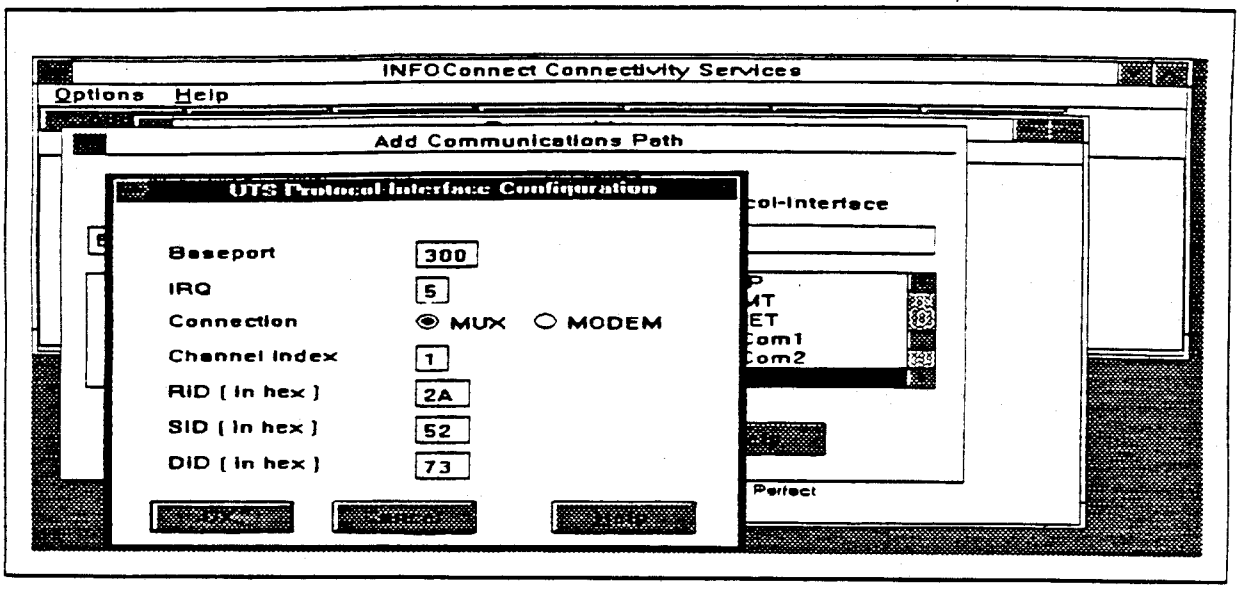
Creating a Communications Path

Visual 1: Depending on the selections made on the Add Communications Path menu, a specific Protocol Interface Configuration menu will appear.

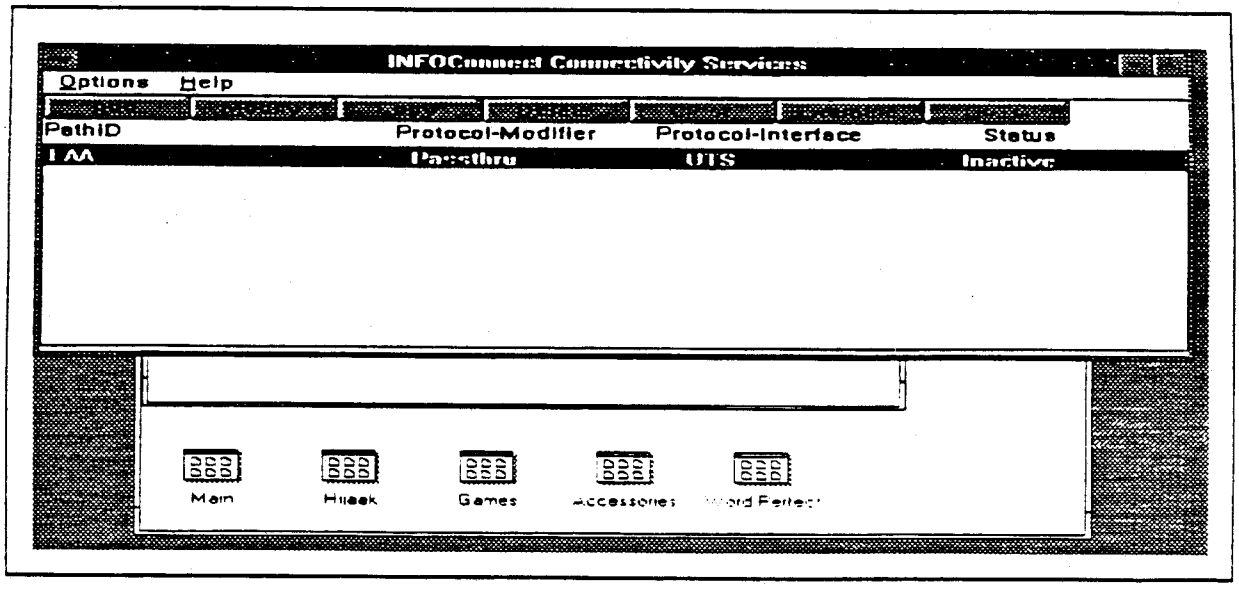
Visual 2: Once this menu is filled out, the new communications path will be added to the INFOConnect Connectivity Services menu.

Creating a Communications Path

- Fill out UTS Protocol-Interface Configuration screen



- INFOConnect screen re-appears with information confirmed



Verify the Connection

Visual 1: We can now verify the communications path by selecting the UTS Emulator icon from the DW windows group.

Visual 2: Designer Workbench will request a PathID and also display a list of other available PathIDs. Select the PathID and double-click on OK.

Note: *Further information on verifying communications paths can be found in Section 6 of the DW Installation and Administration guide. This section covers topics such as:*

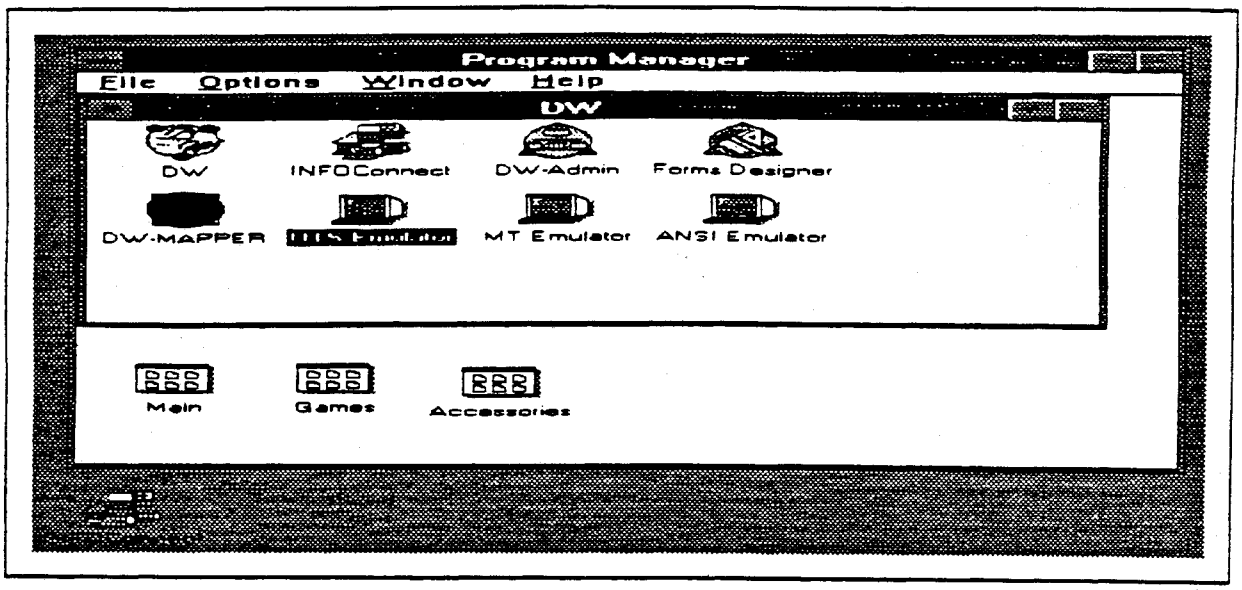
Choosing the right terminal emulator

Solving communications path problems

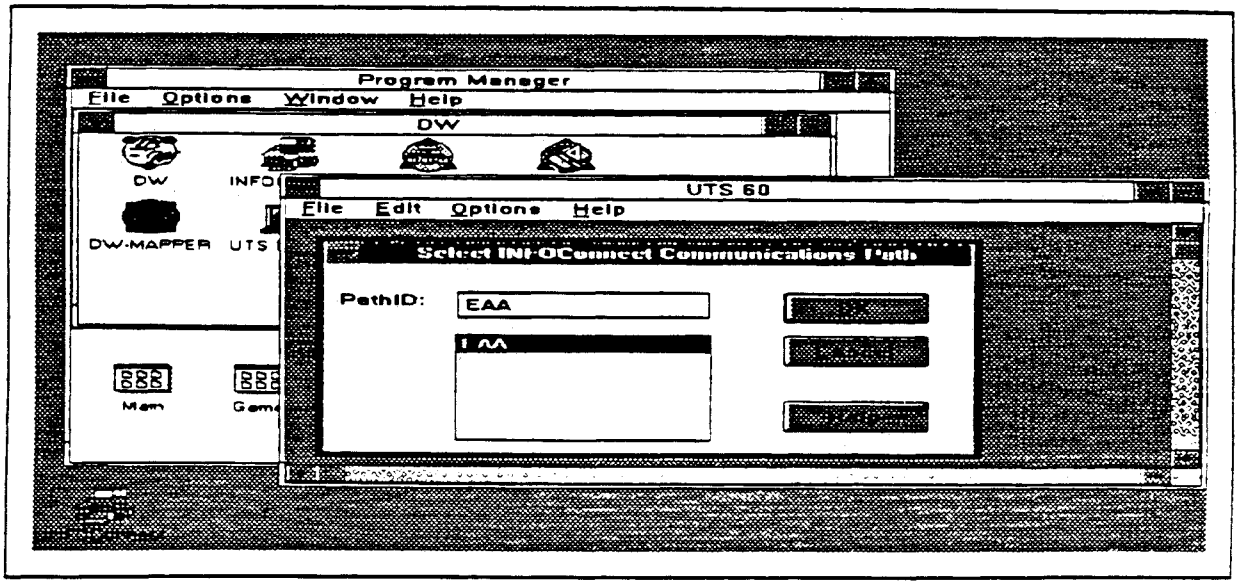
- Checking for workstation hardware-specific problems
- Checking for CONFIG.SYS-related problems
- Checking the host configuration

Verify the Connection

- Select UTS Emulator icon from DW main window



- Select path with UTS 60 window displayed



Verify the Connection

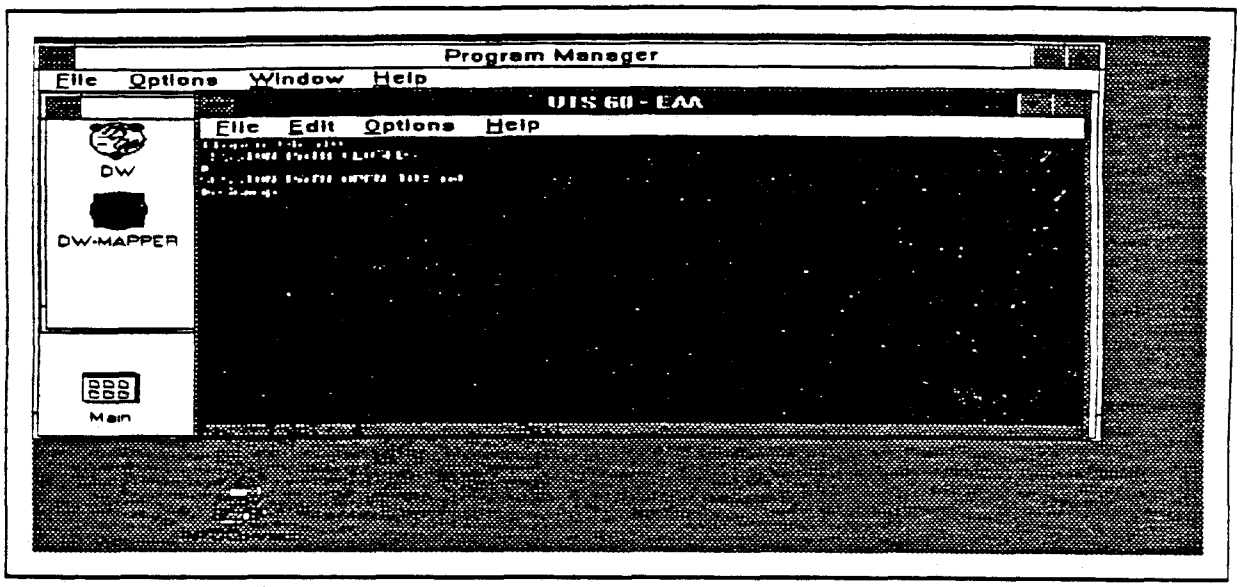
Visual 1: A UTS 60 emulator window appears. At this point in the example contact is made to the host. (Look for a POLL at the bottom of the emulator window as a clue that contact was made.) In this example, we are attempting to access a MAPPER called E36MAP on T4EA10.

Visual 2: The MAPPER idle logo is successfully displayed.

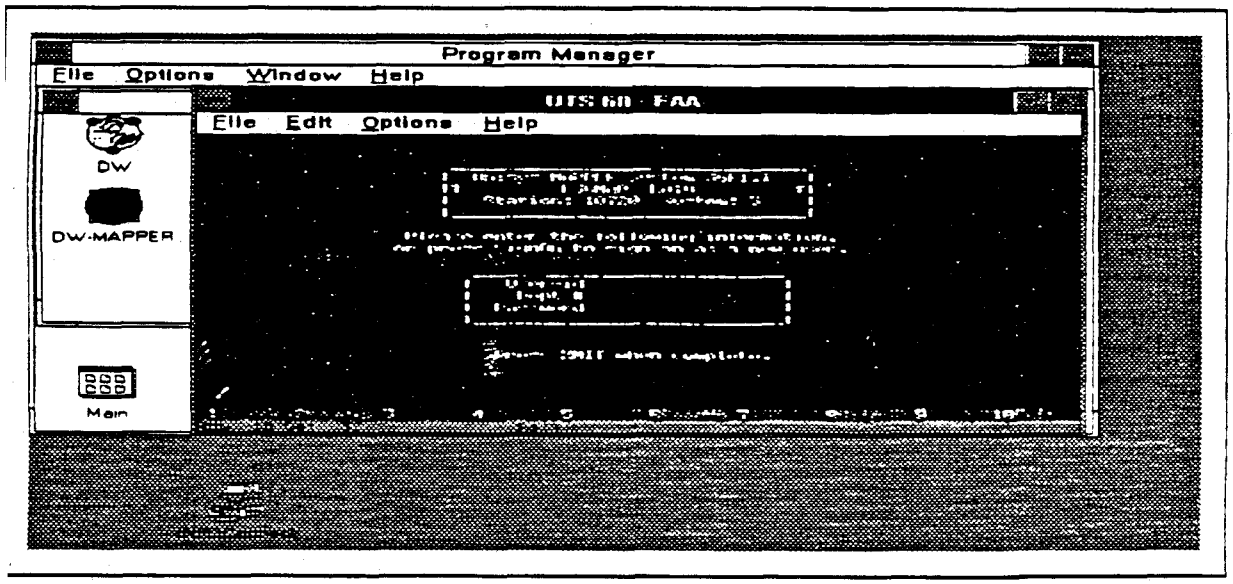
Remember that because we are working in a Windows environment, this emulator window could be maximized or minimized to an icon.

Verify the Connection

- **UTS 60 - EAA window appears - $$OPEN call is attempted**



- **MAPPER idle logo is displayed**



Scripts

Scripts are necessary in order to use MAPPER and LINC APIs.

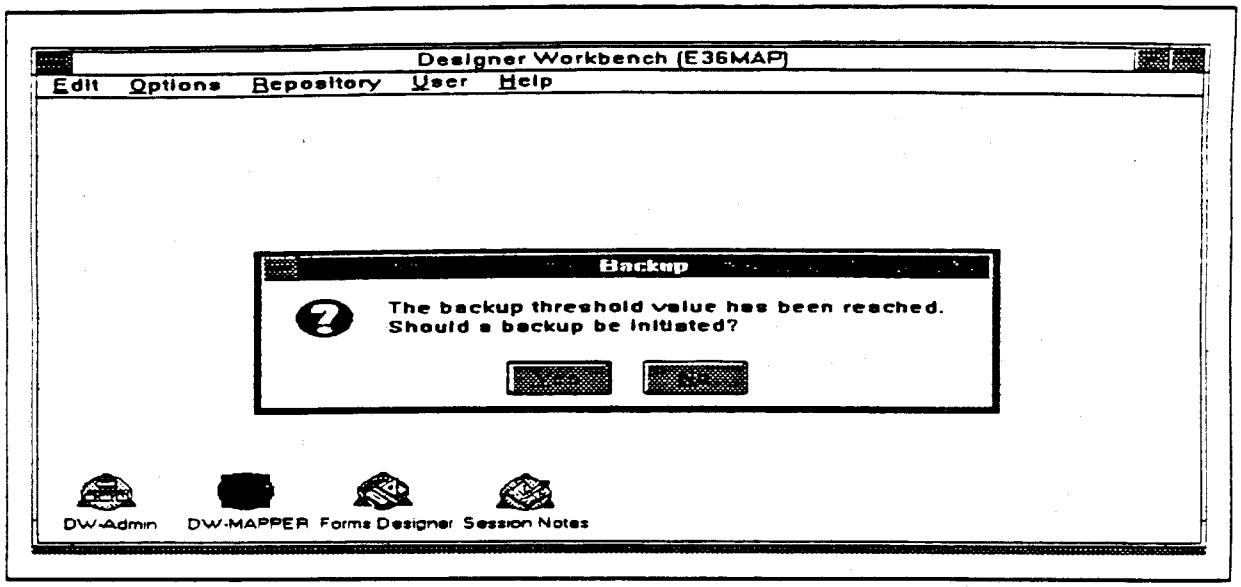
Note: *Information on creating scripts can be found in Section 5 of the DW Installation and Administration guide.*

After worksheets are filled out, the information that was supplied on them will be entered on templates in Designer Workbench.

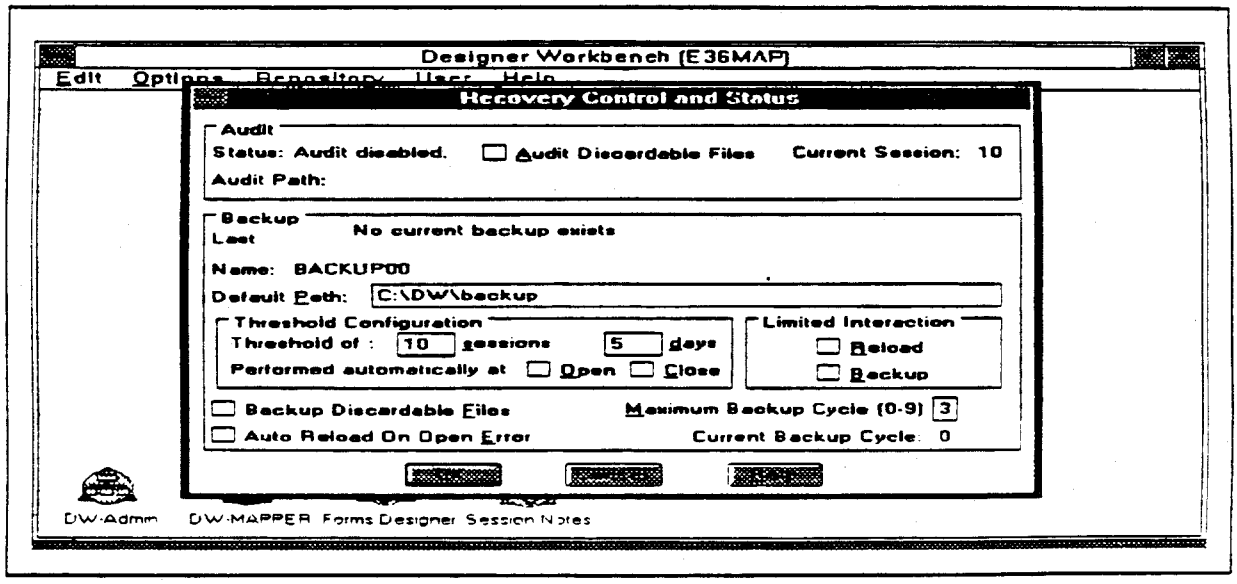
Note: *Script worksheets can be found in Section 2 of the DW Installation and Administration Guide*

Recovery Control and Status

- **Backup/audit message window**



- **Recovery Control and Status screen**



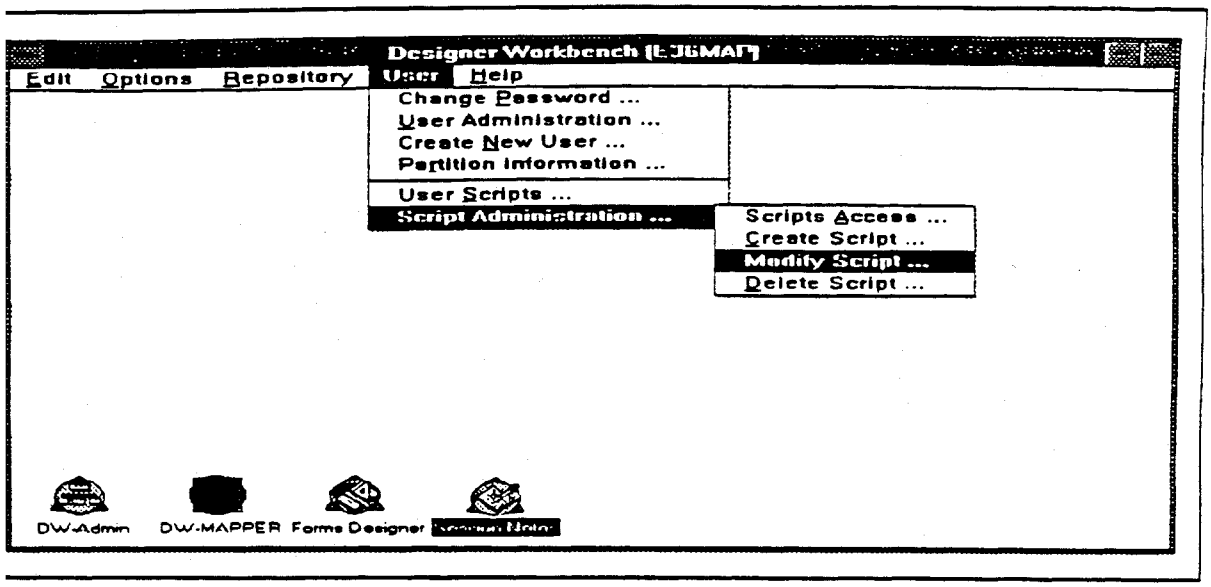
Recovery Control and Status Options

Once again, in addition to providing information concerning Repository backup and auditing, the Recovery Control and Status screen provides the opportunity to customize the procedure. The visual on the opposite page lists some of the things that you can determine by updating the screen.

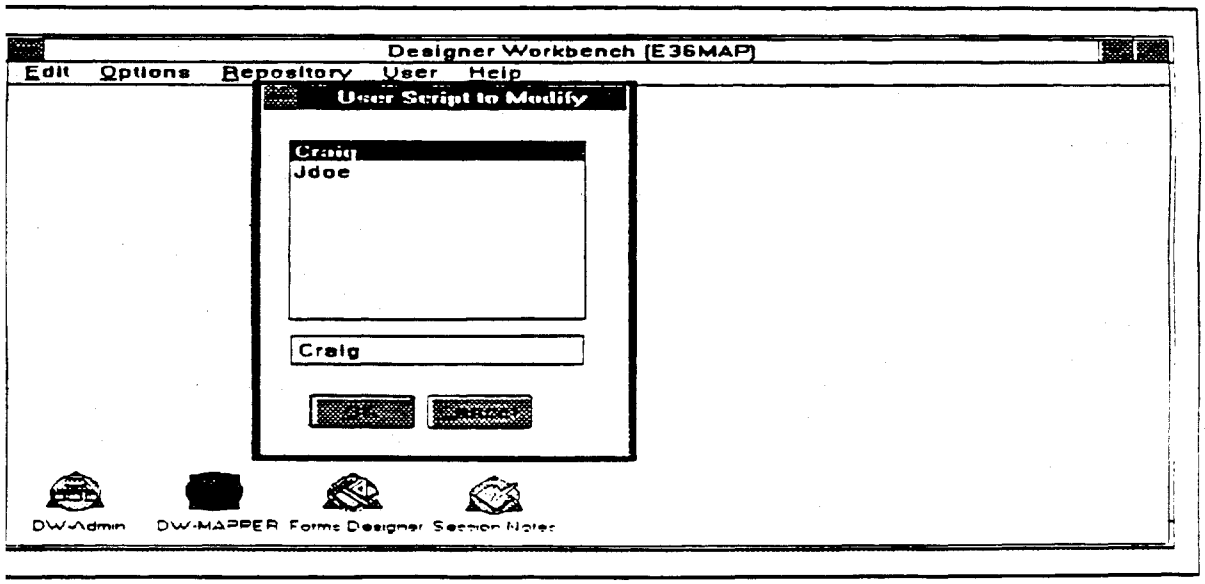
Note: *Refer to pages 8-6 through 8-12 of the DW Installation and Administration Guide for explanations of each task.*

Modifying Scripts

- **Select Modify Script**



- **Select script**



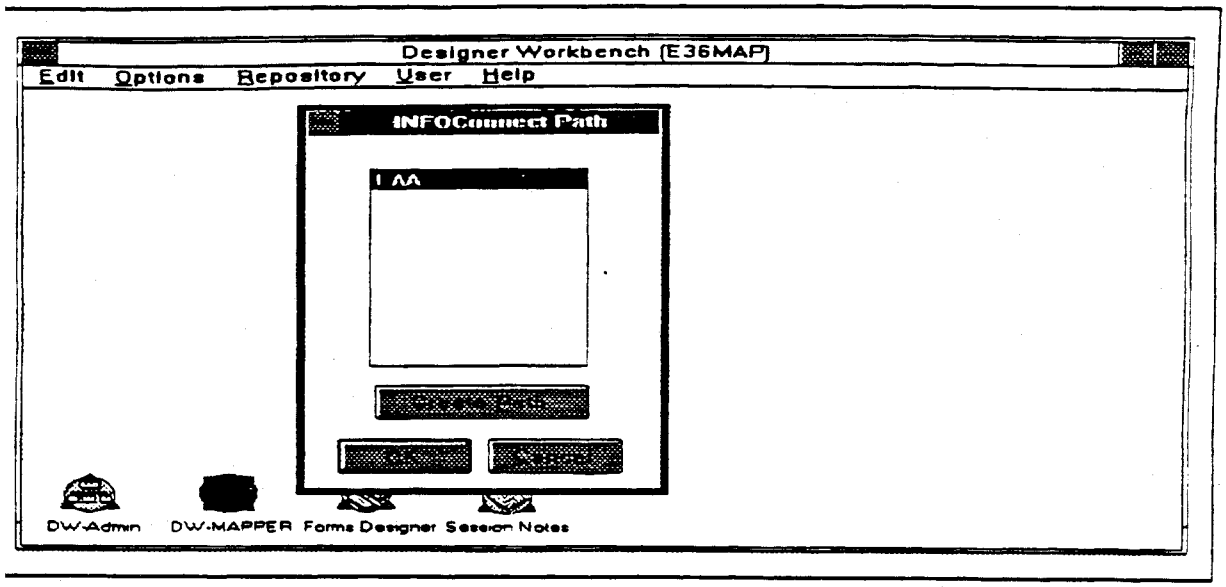
Modifying Scripts

Visual 1: The INFOConnect Path screen is displayed. This menu will contain a list of all the paths created on the Designer Workbench. In this example, only one path has been created. Select the path and select OK.

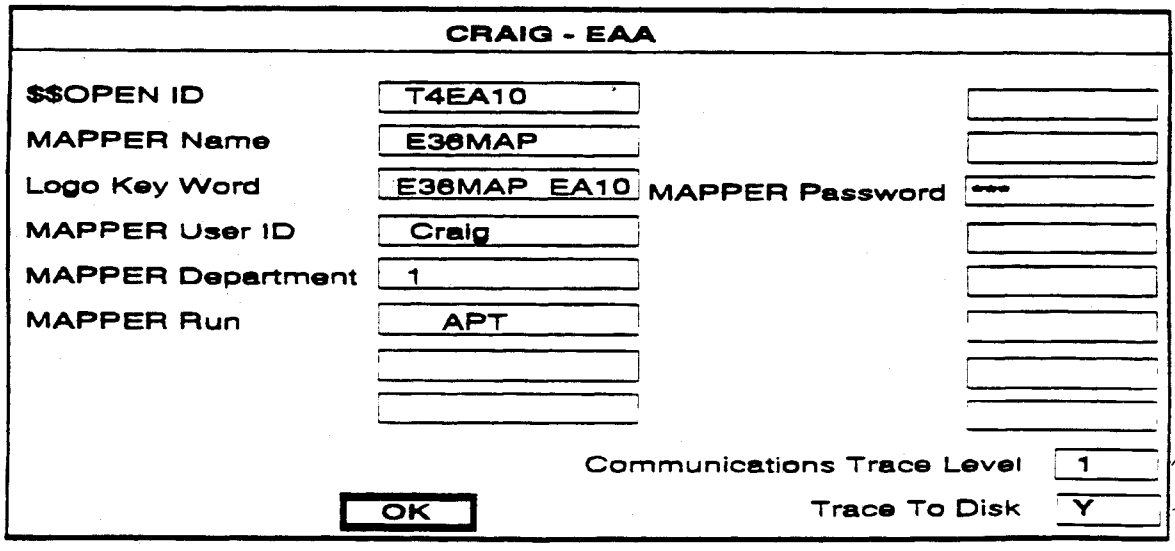
Visual 2: The script template is then displayed with the entries made when the script was created. Field entries can now made or changed. In this example, an addition has been made to the script template. The APT MAPPER run will be started whenever the script is processed.

Modifying Scripts

- Select INFOConnect path



- Change the script and select OK



Modifying Script Access for Users

When script access for users is changed, the Connection Scripts menu is once again used. You may recall that this is the screen that was used to initially distribute script access.

Visual 1: To modify script access for users, select 'User' from the menu bar, select 'Script Administration', select 'Script Access', highlight the user to change.

Visual 2: Select the user script. Select 'Delete Access'.

Note: *For more information and procedures on Modifying Script Access, refer to pages 10-5 and 10-6 of the DW Installation and Administration Guide.*

Reading and Writing Files

- **Read from DOS file**
  - Initially copies files from DOS into the Repository
  - Files are read into current working partition
- **Write to DOS file**
  - Copy an object in the Repository to a DOS file
  - Object can be used by other products
- **Similar but different than Import/Export functions**
  - Import/Export used (together consecutively) to copy Repository-formatted objects in and out of Repository
  - Read/Write used to copy information into or out of DOS format files outside a Repository
- **File Read and Write**
  - BMP (bitmapped images in BMP format)
  - Data (unformatted data)
  - PCX (bitmapped images in PCX format)
  - MAPPER Graphics (in MGL format)

Read from DOS Example

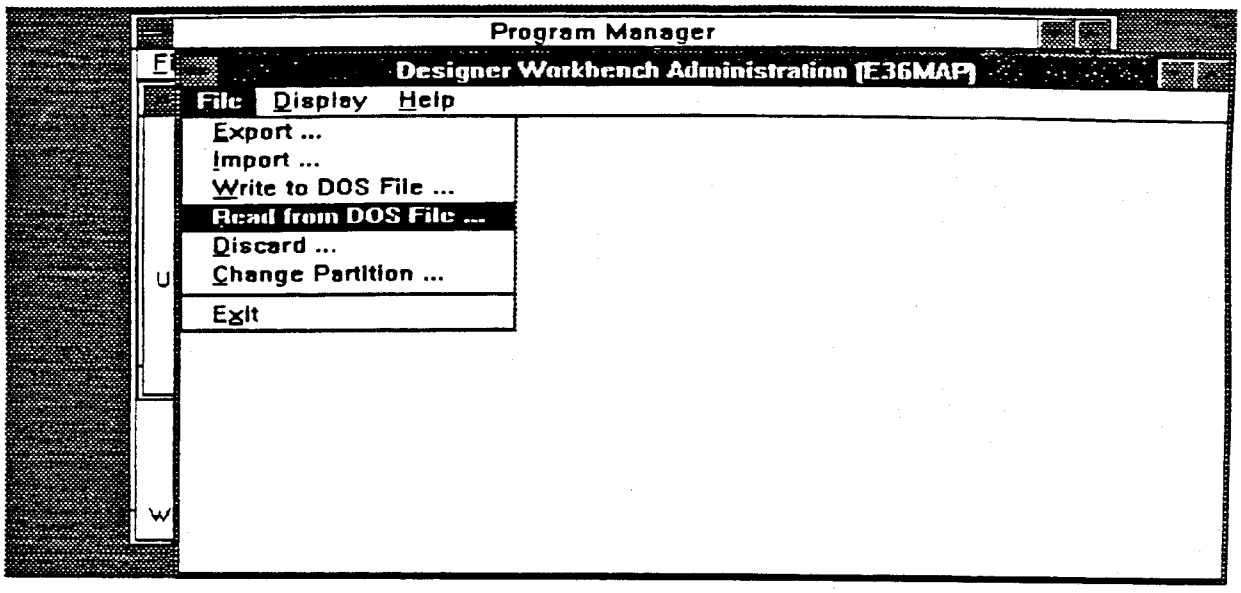
The examples on the following pages illustrate the 'Read from DOS' procedure. In this example we wish to read a BMP file called C:\WINDOWS\BOXES.BMP into the current partition (E36MAP).

Visual 1: To Read a File from DOS, select 'File' from DW Administration, select 'Read from DOS File'.

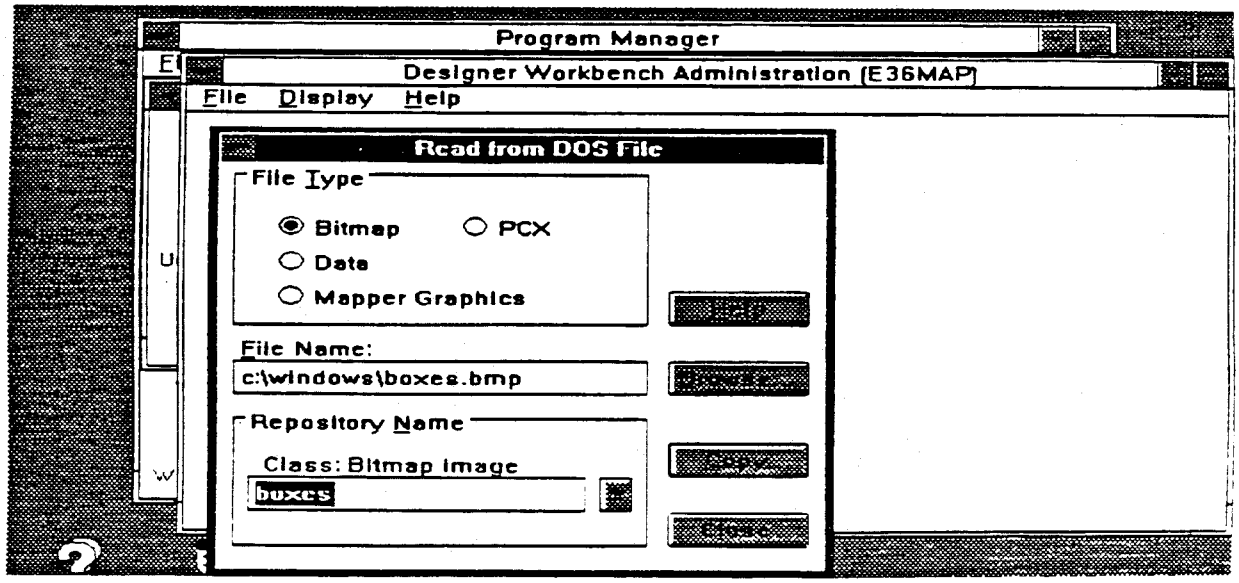
Visual 2: Select File Type, enter File Name to Read From, enter name to represent the file (object) retrieved.

Read from DOS Example

- Select 'File', select 'Read from DOS file'



- Select file name to retrieve, enter object name



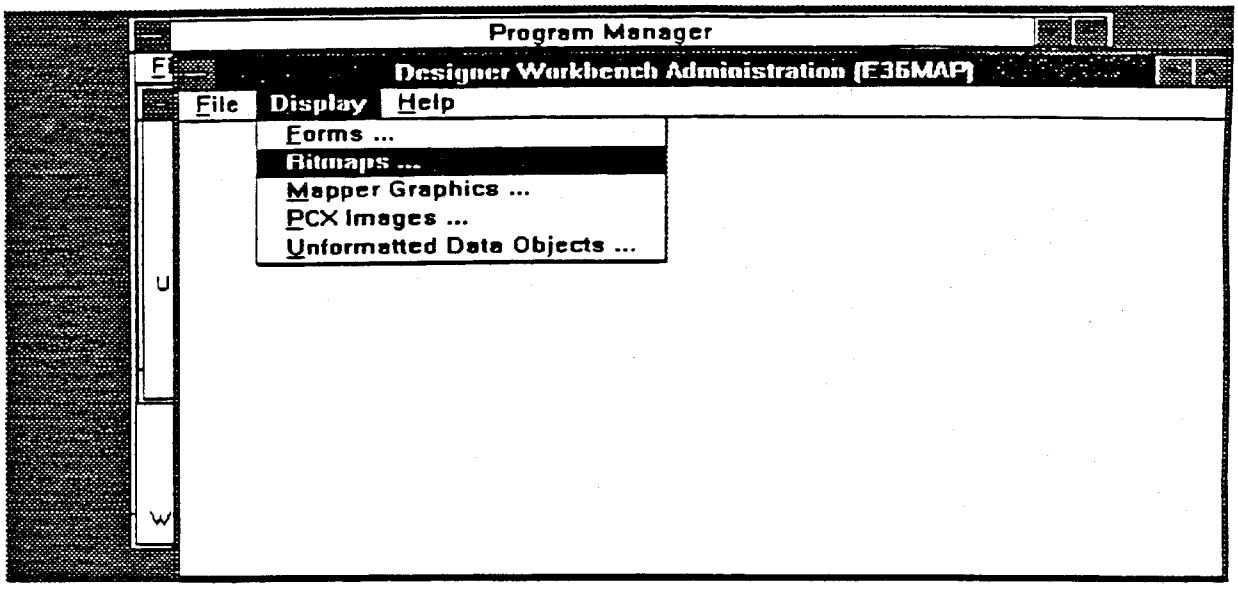
Read from DOS Example

Visual 1: To verify the Read function, select 'Display' from DW Administration, select 'Bitmaps'.

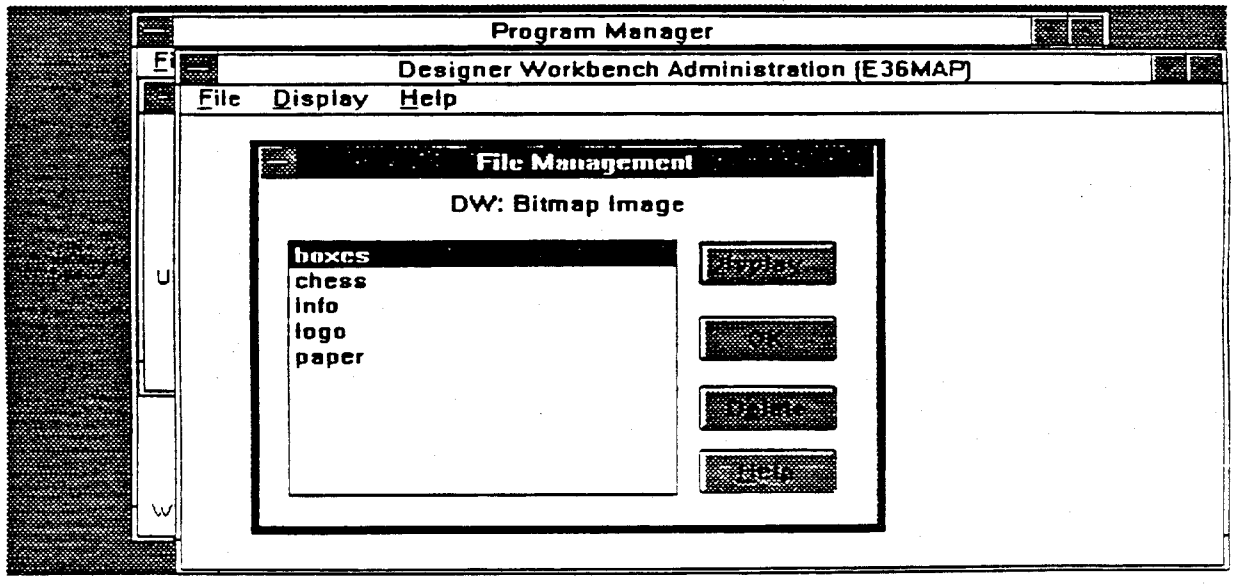
Visual 2: 'Boxes' now is listed with the other Bitmap images in the E36MAP Repository partition.

Read from DOS Example

- Select 'Display', select 'Bitmaps'



- 'Boxes' is verified



Write to DOS File Example

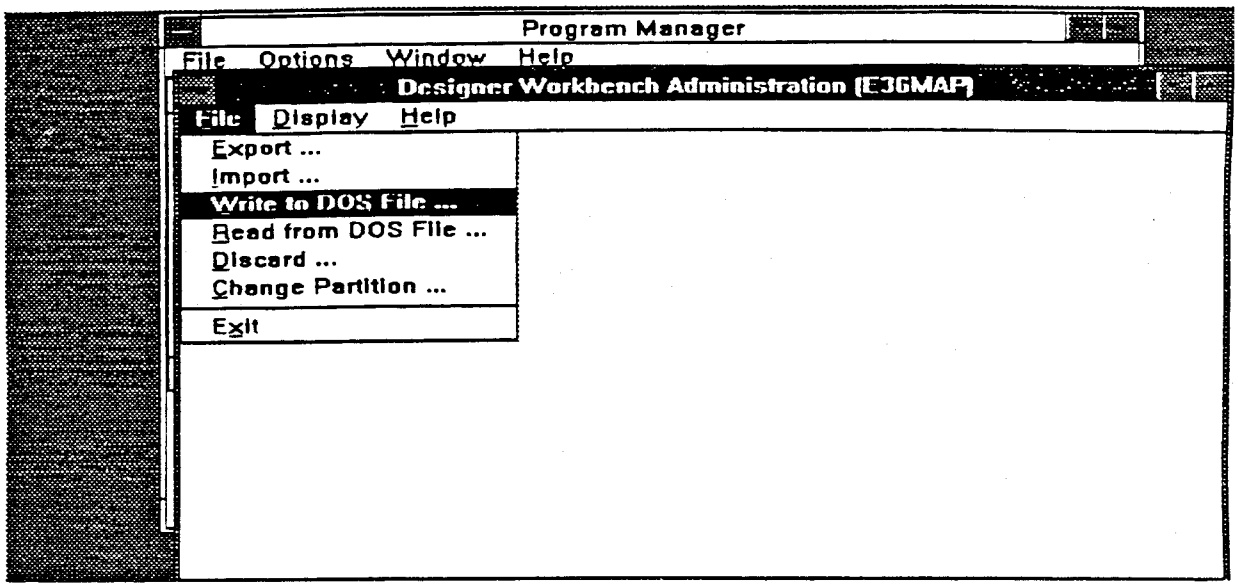
In this example, we wish to take the object called C:DW\BIN\LOGO.BMP and write it to the DOS file C:HOLDIT.

Visual 1: Begin by selecting 'File' from DW Administration. Select 'Write to DOS File'.

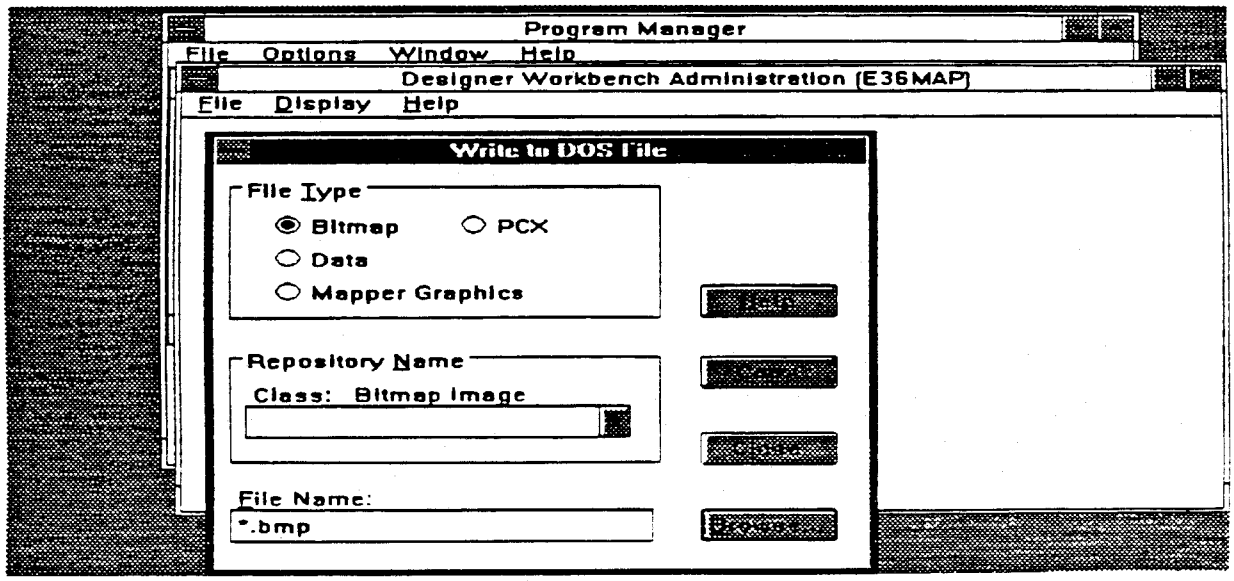
Visual 2: Write to DOS menu appears.

Write to DOS File Example

- Select 'File', select 'Write to DOS File'



- Write to DOS File menu displayed



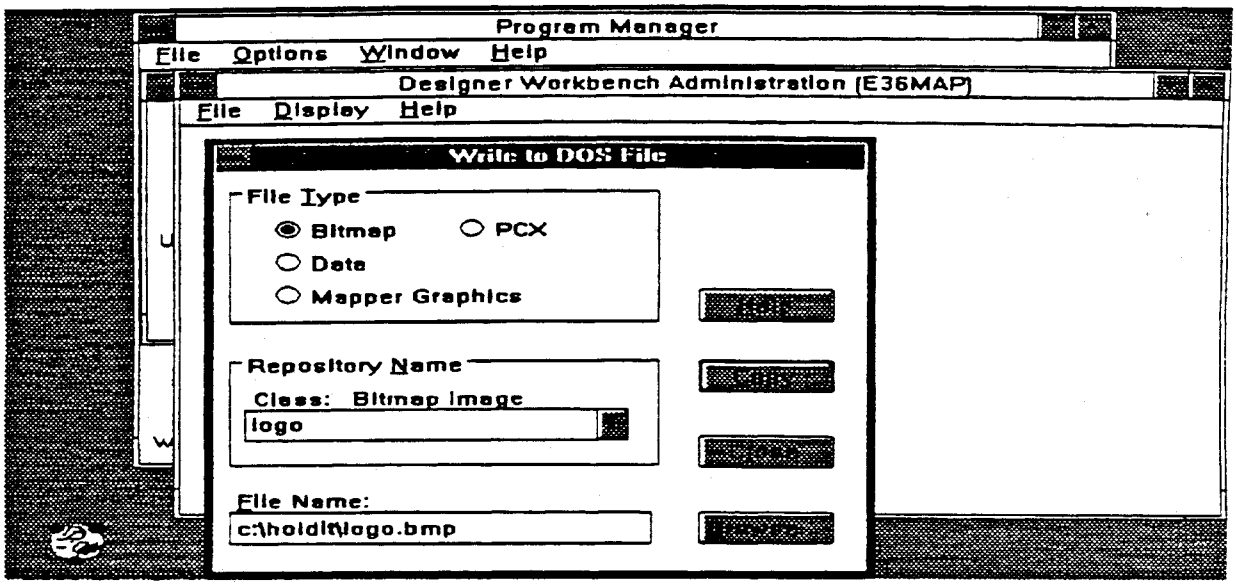
Write to DOS File Example

Visual 1: Select the File Type to write. Enter the object name to write. Enter the receiving file name.

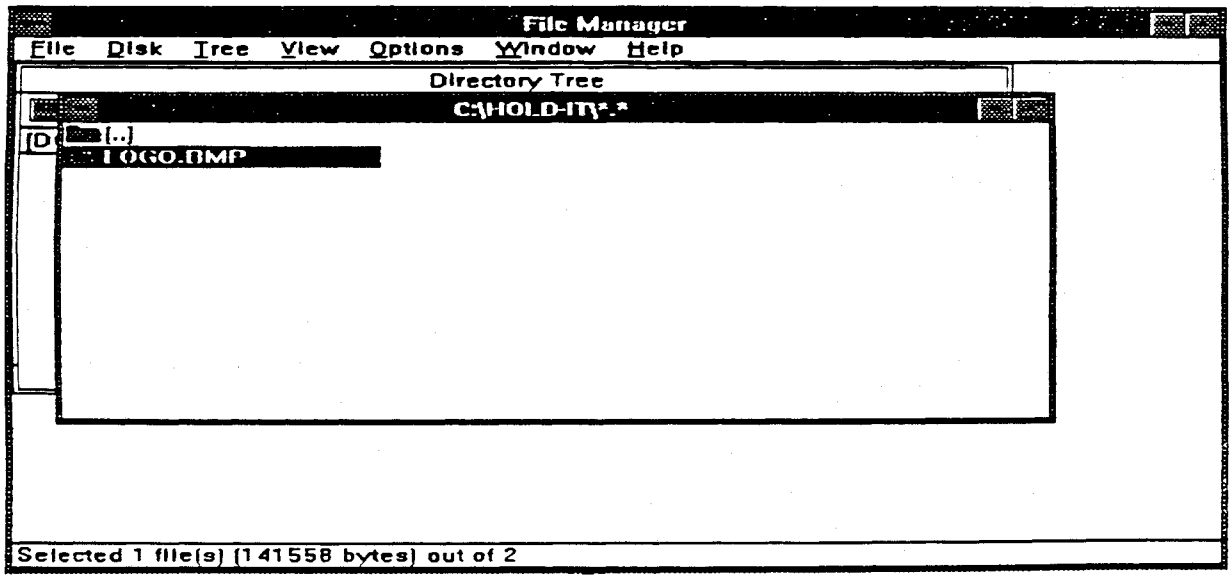
Visual 2: Once again, go to the Windows File Manager to verify the written file.

Write to DOS File Example

- Enter Repository item to copy and receiving file, select Copy



- File is verified in Windows File Manager



Objectives

Upon completion of this module, you should be able to

1. Describe the features of the Forms Designer
2. Describe the benefits of using the Forms Designer with
   a. MAPPER applications
   b. LINC applications
3. Use the Designer Workbench Developer Training Guide and online help to obtain additional information on the Forms Designer
4. Create a form with the Forms Designer

Learning Sequence

This module provides an overview of the Designer Workbench Forms Designer. The topics to be covered are listed on the opposite page.

Section 3 of the Designer Workbench Developer Training Guide (7831 9738-00) is a good source of information when learning to use the Forms Designer. The material is presented in a step by step 'training' style approach. Therefore, rather than duplicating the information in the Developer Training Guide, this module will provide an overview of the Forms Designer and how to use the Training Guide and online help.

The goal of this module is to acquaint the user with the major components of the Forms Designer and provide the information to create and save a form. This module is **not** intended to provide information on how to create a LINC or MAPPER application. For information on creating 4GL applications, refer to the Contents page of the Designer Workbench Developer Training Guide. The following topics are covered in the DW Developer Training Guide:

- How MAPPER and Designer Workbench work together
- Creating MAPPER application forms
- Creating MAPPER application data files and runs
- Testing MAPPER applications
- MAPPER form run statements
- Using the Forms Designer with LINC software

References

Documentation referenced in this module:

Designer Workbench Developer Training Guide (7831 9738-000)

Learning Sequence

- **Overview of the Forms Designer capabilities**
- **Describe benefits of the Forms Designer as they relate to**
  - MAPPER
  - LINC
- **Discuss major components of the Forms Designer**
- **Utilize hard copy and online documentation**
- **Exercise - create and save a form**

Forms Designer

The DW Forms Designer is similar to the Windows Paintbrush facility. Text, fields, and other objects are created by selecting tools and using the mouse. No coding of any kind is necessary to create a form. Forms can be easily modified and saved into files.

Forms Designer

- Interactive Designer Workbench forms-design tool
- Component of DW used to create and modify forms for MAPPER and LINC applications
- Windows application allowing use of mouse and clipboard
- Uses Paintbrush-like tools to create and modify objects

Forms Designer and MAPPER

We have already seen how MAPPER, running in the Windows environment, has changed in functionality and appearance. Another major change to MAPPER is the use of input/output forms (screens) created with the Forms Designer. MAPPER users executing runs which display DW forms now have:

- Graphics and pictures
- Mouse instead of using arrow and transmit keys
- Information selected by using pull-down bars, list boxes, and buttons

Forms Designer and MAPPER

- **Create object-oriented input/output forms used in MAPPER runs**

- **Buttons and button groups**
  - User can click on them to perform actions

- **List boxes**
  - List multiple items from which user can choose
  - Item information is loaded into list boxes from MAPPER reports

- **Fields**
  - Input fields for data entry/output fields for displaying information

- **Images**
  - BMP or PCX format files enhance the appearance of forms

- **Text**
  - Variety of colors, sizes, and fonts

- **Pull-down menus**

Forms Designer and MAPPER

Note: *For more information on the Forms Designer and MAPPER refer to pages 2-2 through 2-8 of the Designer Workbench Developer Training Guide.*

Forms Designer and MAPPER

- **Work together to download and upload information to run applications**
- **Form objects have a data order for passing input to host**
- **Multi-copy forms are identified via version control**
  - Application defined
  - Null version
  - Timestamped
- **Optionally transfers and stores forms and related objects in a MAPPER report**
- **MAPPER run statements are also available for creating forms**
  - WBX run contains examples

Forms Designer and LINC

The DW Forms Designer can be used to customize and improve the appearance of a LINC screen. You can use the Forms Designer with LINC software to:

- Replace fields with list boxes
- Replace fields with buttons or button groups
- Change the size and type of font
- Change the color of text
- Change the physical location of fields and objects on the screen

You cannot use the Forms Designer with LINC software to:

- Add menus
- Add or edit fields

Forms Designer and LINC

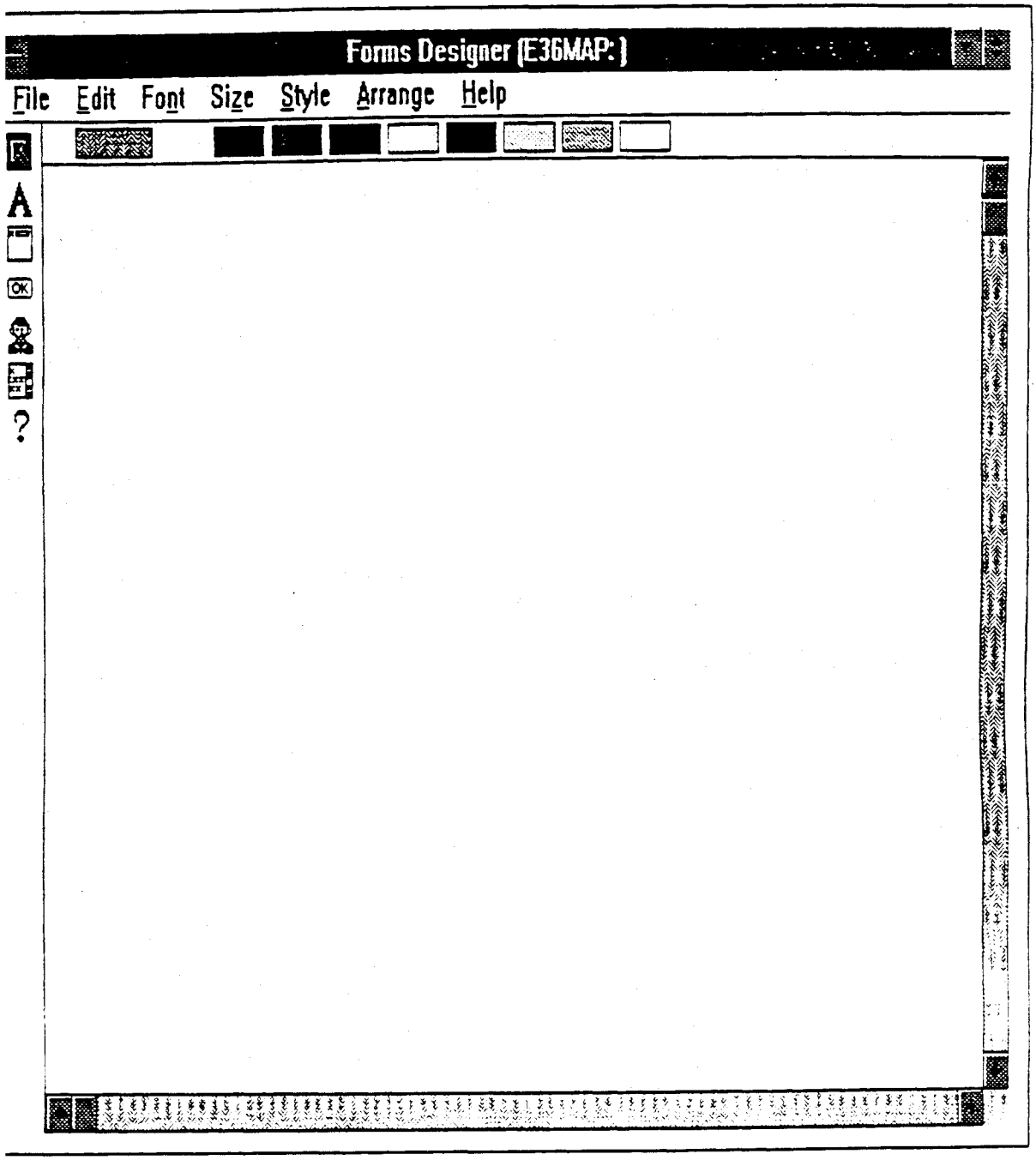
- **Customize and improve the appearance of LINC screens**
  - Replace fields with list boxes
  - Replace fields with buttons or button groups
  - Change type and size of fonts
  - Change the color of text
  - Change the physical location of fields and objects on the screen
- **It is not possible to**
  - Add menus
  - Add or edit fields

Forms Designer

The Forms Designer window is shown on the opposite page. This window is the work area for creating forms. It is made up of three major components:

- Tools
- Menu bar
- Color palette

Forms Designer Window



Specifying Form Characteristics

The first step in creating a form is completing the Specifying the Form Characteristics screen. It is on this menu that you define certain characteristics of the form being created. The following characteristics can be defined:

- Form title
- Size of the form (in logical inches)
- Application help file location (MAPPER only - the name of the MAPPER report that contains online help for the host application)

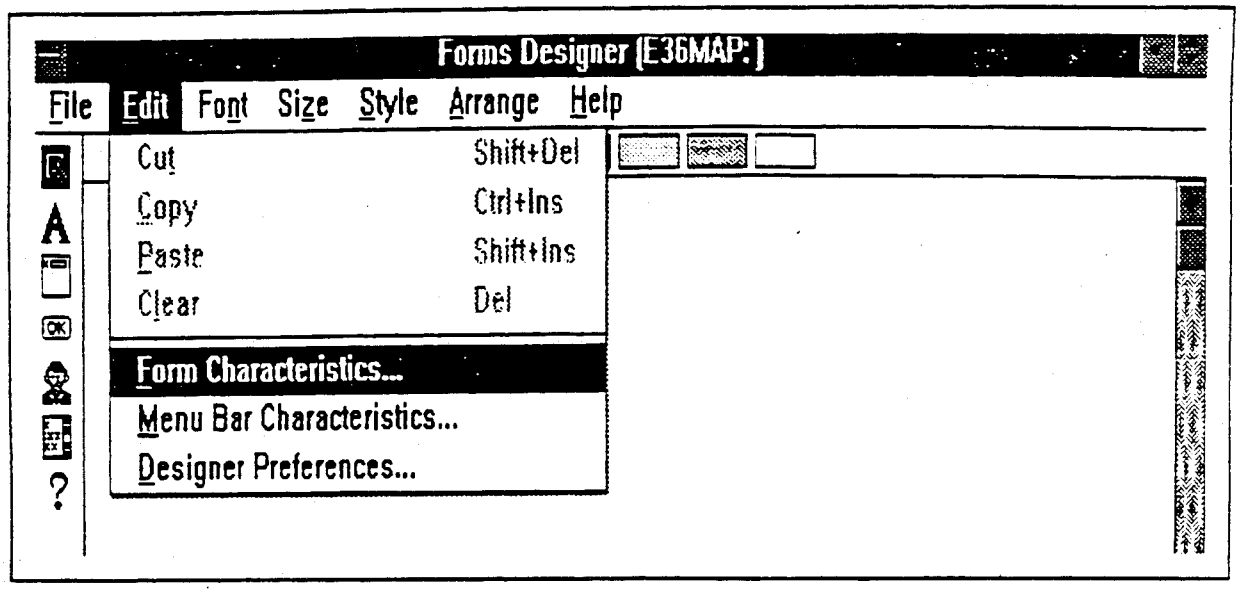
Visual 1: Select 'Edit' from the Forms Designer menu bar. Select 'Form Characteristics'.

Visual 2: The Form Characteristics menu is displayed.

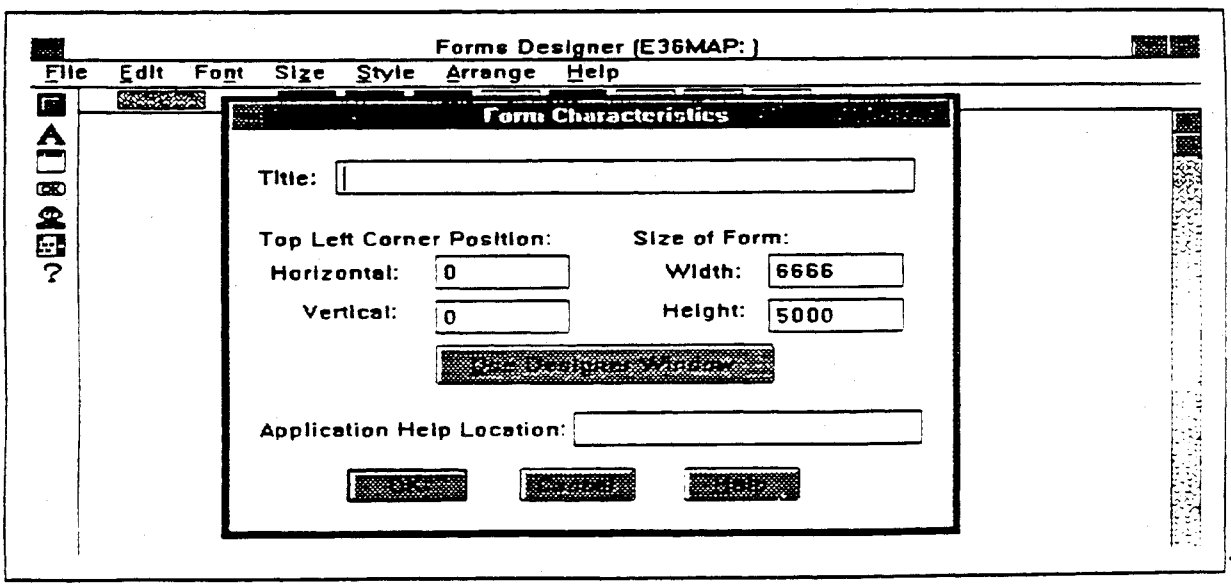
Note: *For more information concerning Specifying Form Characteristics, refer to pages 3-6 and 3-7 of the DW Developer Training Guide.*

Specifying Form Characteristics

- **Select 'Form Characteristics'**



- **Form Characteristics menu**



Using Online Help

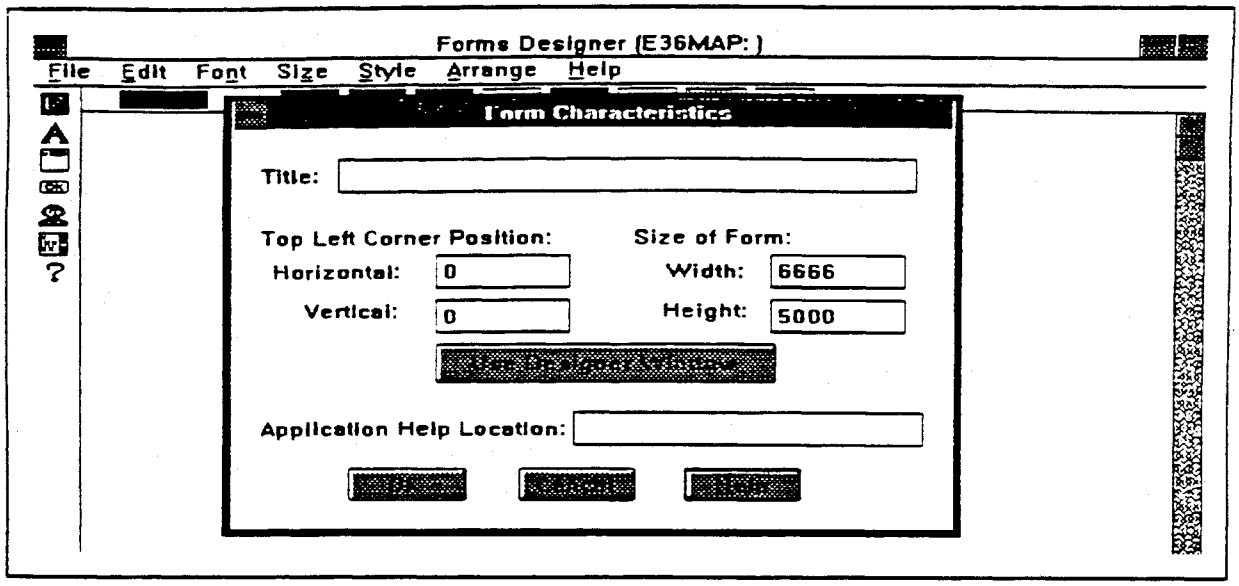
Even though the DW Developer Training Guide provides a great deal of information for creating forms, additional information exists in online help files. In this example we wish to Specify Form Characteristics. Pages 3-6 through 3-8 provide general information on this topic. But suppose we want to obtain more specific field information for the Form Characteristics (or any other) menu screen? The solution to this problem is to use online help.

Visual 1: In this example, we wish to obtain information on the Use Designer Window box. With the Form Characteristics screen displayed, click on the help button in the lower right hand corner of the window.

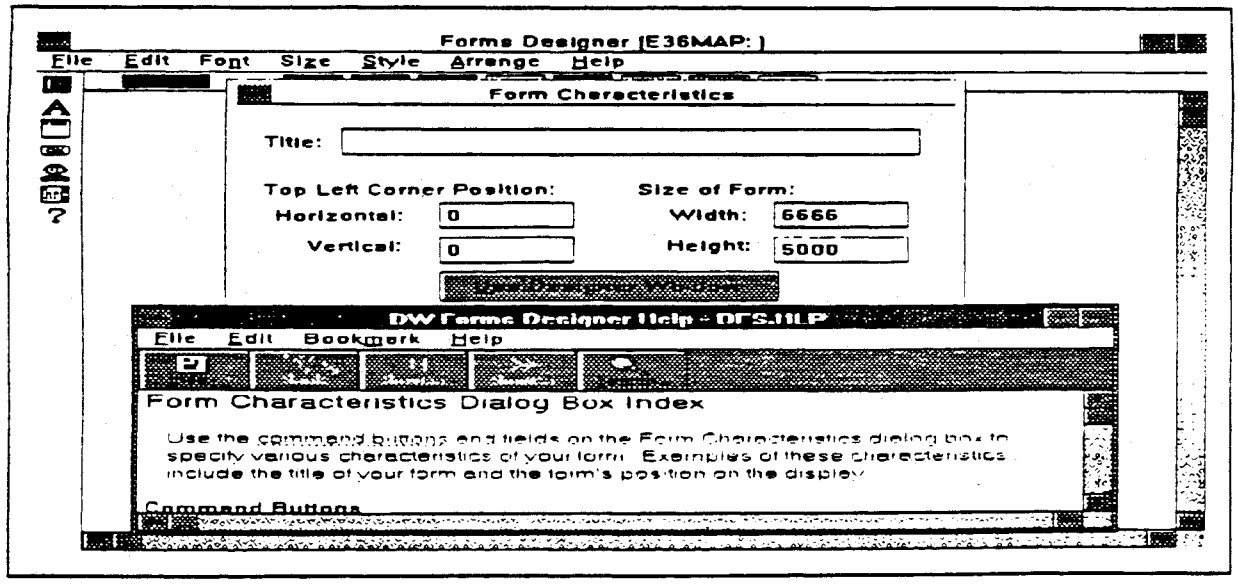
Visual 2: The DW Forms Designer Help window is displayed.

Using Online Help

- **Select Help button from Form Characteristics menu**



- **Help window is displayed for Form Characteristics fields**



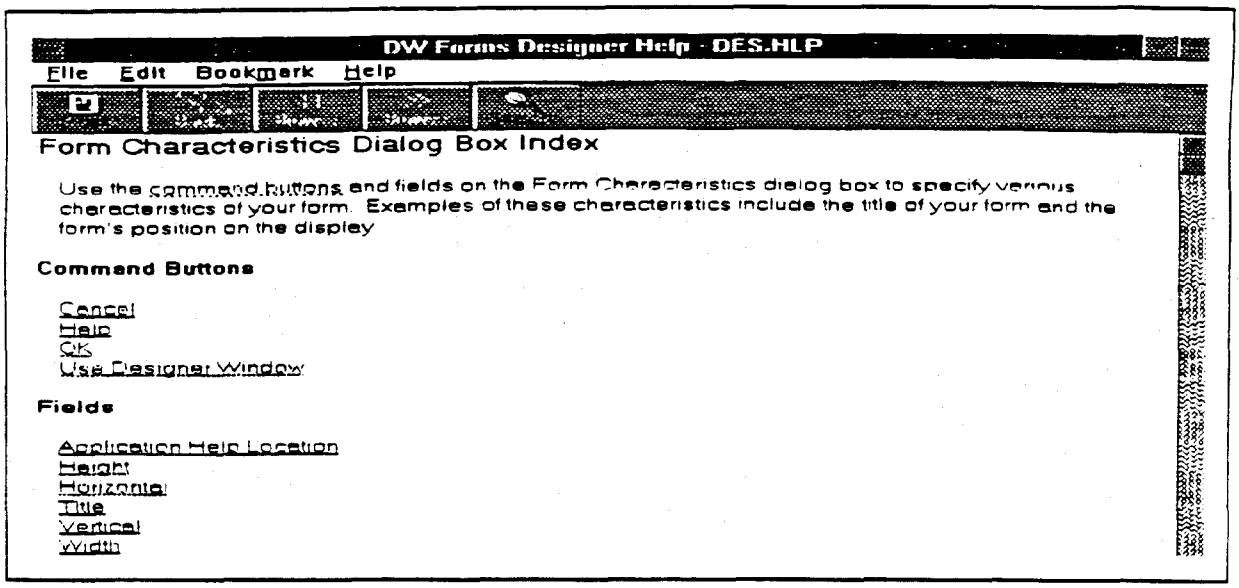
Using Online Help

Visual 1: The help window has been maximized. Now, roll through the window to find the topic on which to receive specific help. Since we wish to receive help on the Use Designer Window button, we position the cursor to the appropriate position of the help screen and click the mouse.

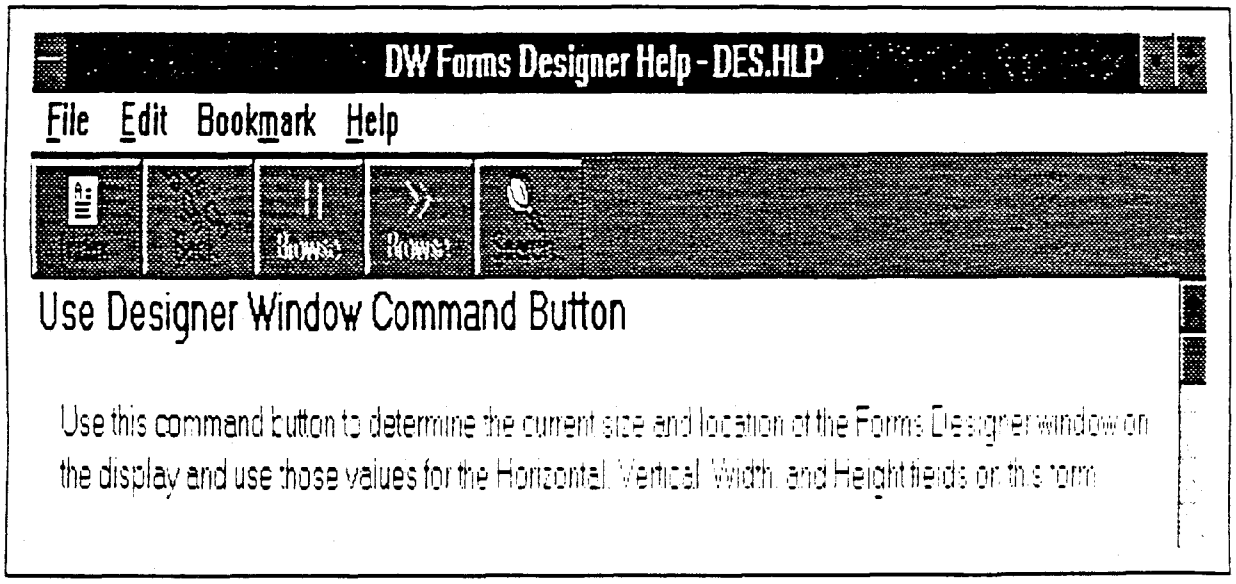
Visual 2: A new help window is displayed with the requested information.

Using Online Help

- **Window is maximized, select 'Use Designer Window'**



- **Specific help window is displayed**



Menu Bar Characteristics Option

You can design a menu bar for the form to be created by using the Menu Bar Characteristics option. The menu bar will be similar to all other Windows menu bars in functionality. You have the ability to define the following characteristics:

- Name on the menu bar
- Associated options (choices) for each menu
- Label (name) for the menu items
- Identification the application should use
- Menu item characteristics

Even though the menu bar is created and defined, it cannot be viewed until utilized in an application.

The application using the form will determine the logic of the menu bar selections.

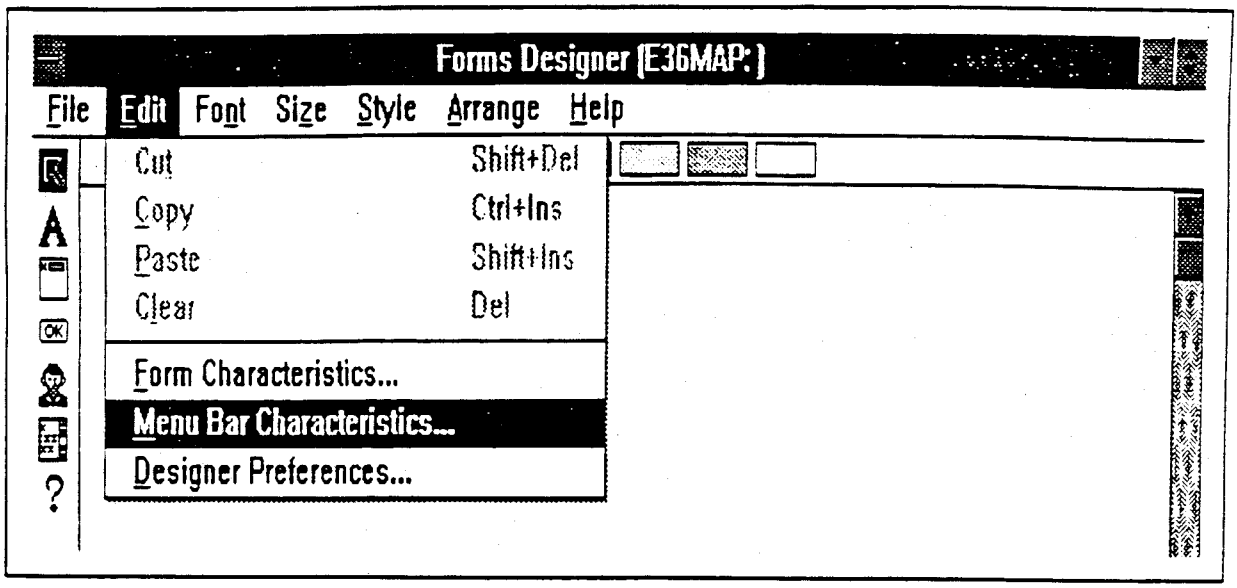
Visual 1: Select 'Edit', select 'Menu Bar Characteristics'.

Visual 2: The Menu Bar Characteristics screen is displayed. Complete the screen and use Help button for more information on specific fields.

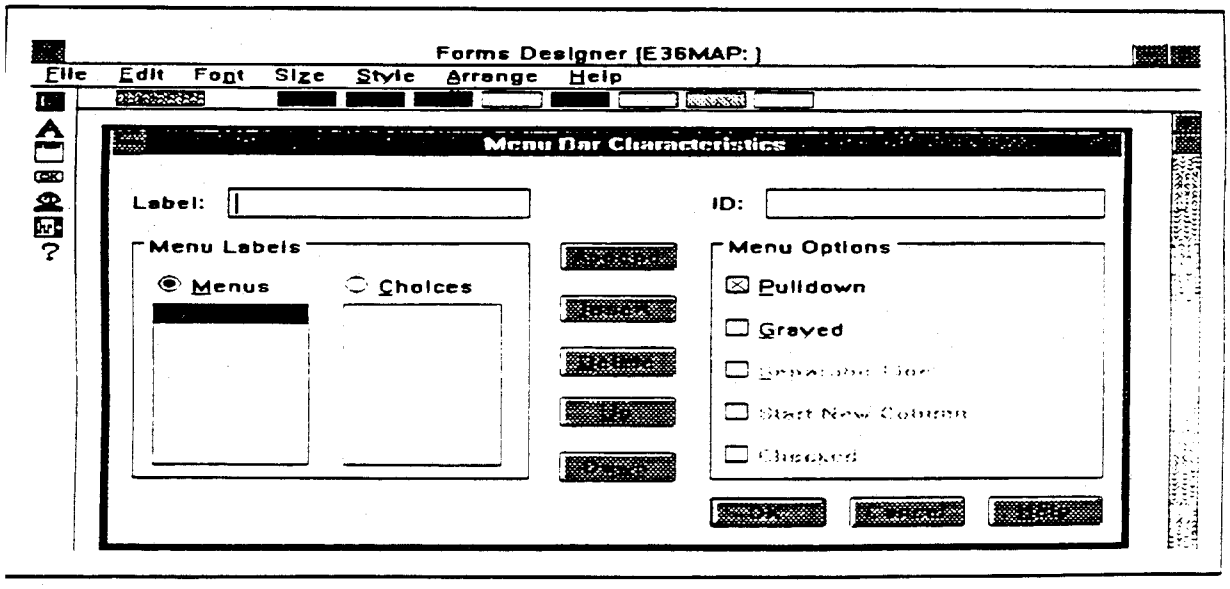
Note: *For more information on the menu bar characteristics option, refer to pages 3-7 and 3-8 of the DW Developer Training Guide.*

Menu Bar Characteristics Option

- **Select 'Menu Bar Characteristics'**



- **Menu Bar Characteristics screen**



Forms Designer Tools

The DW Forms Designer is made up of seven different tools which correspond to specific functions. The seven tools available are:

- Pointer
- Text
- Field
- Button
- Image
- List
- Help

Icons representing these tools can be found along the left hand side of the Forms Designer work screen. The visual on the opposite page provides the functions for each tool.

Forms Designer Tools

Name	Functions
Pointer Tool	Selects, sizes, moves and edits form objects
Text Tool	Adds text objects
Field Tool	Adds field objects
Button Tool	Adds button objects (command, check and option buttons)
Image Tool	Adds image objects
List Tool	Adds list box objects
Help Tool	Selects context-sensitive help

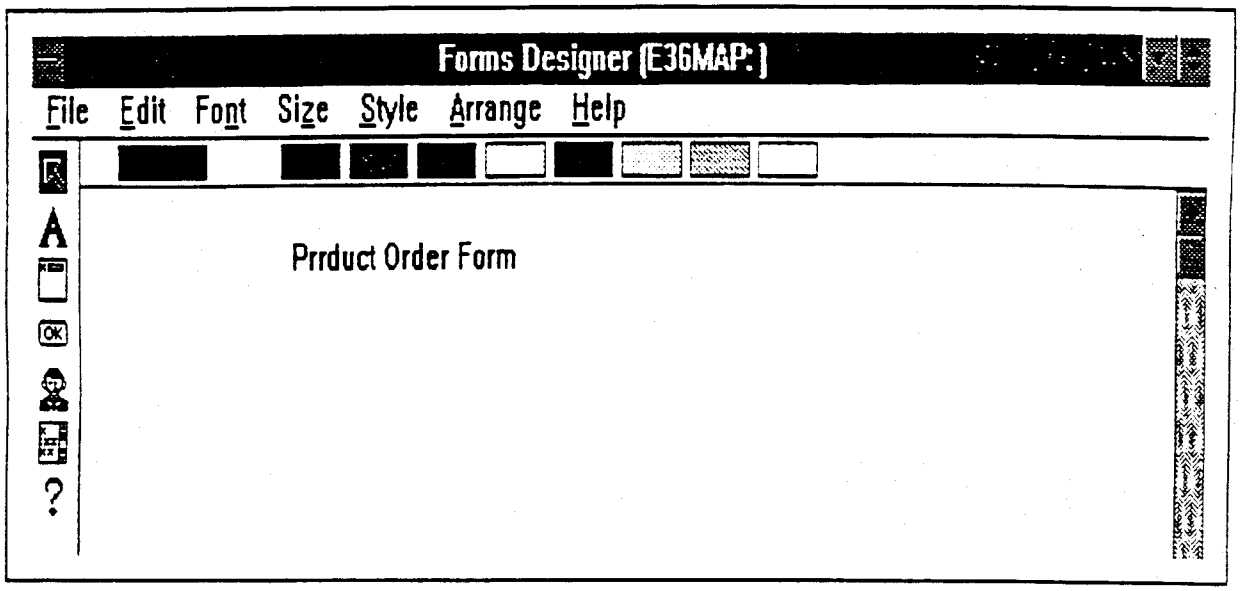
Adding Text

Visual 1: To add text to the form, select the Text icon, position the cursor, and enter text.

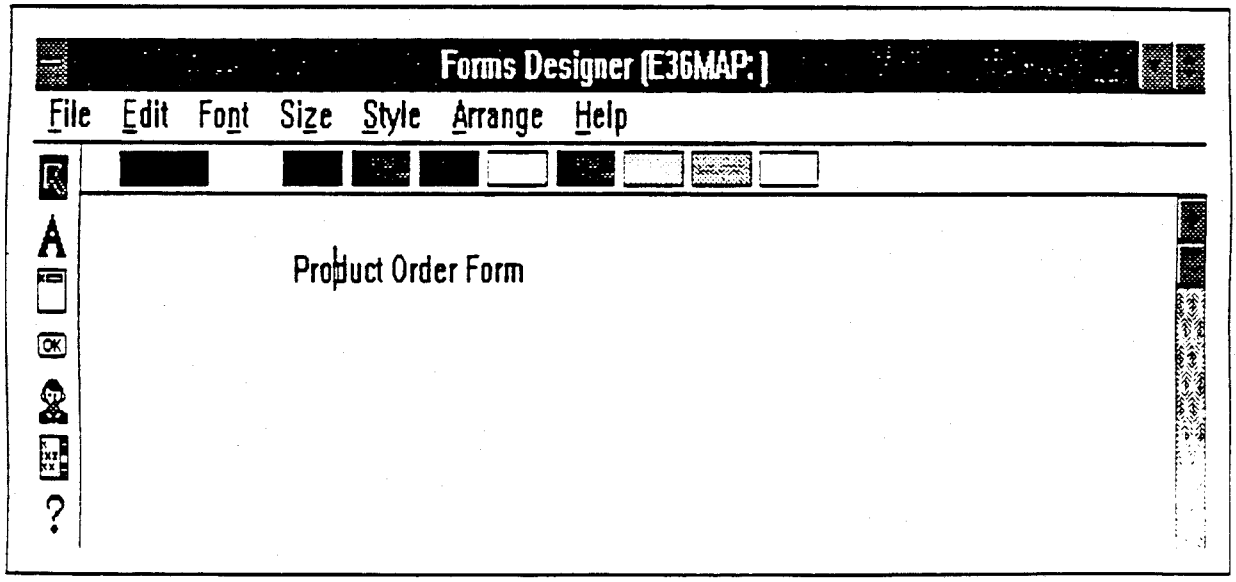
Visual 2: Once text has been entered, it can be edited. Double-click on the text block using the Pointer tool. Edit the text.

Adding Text

- **Select text icon, position cursor, enter text**



- **Highlight text entry with double click, correct typo**



Sizing

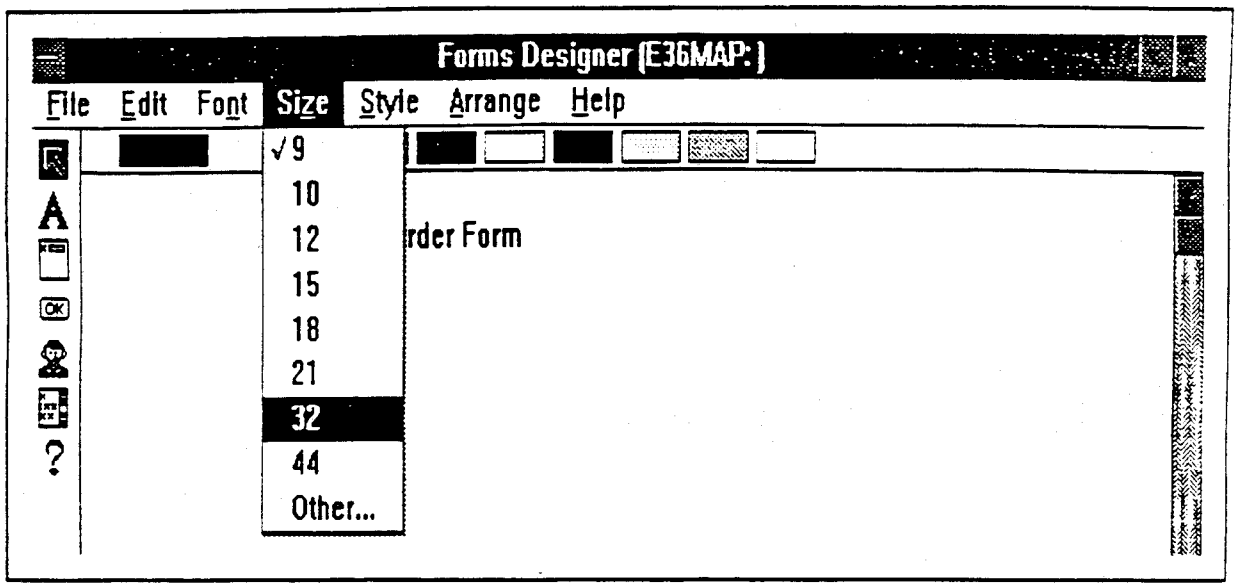
The size of text can easily be changed.

Visual 1: Select 'Size', select the text size.

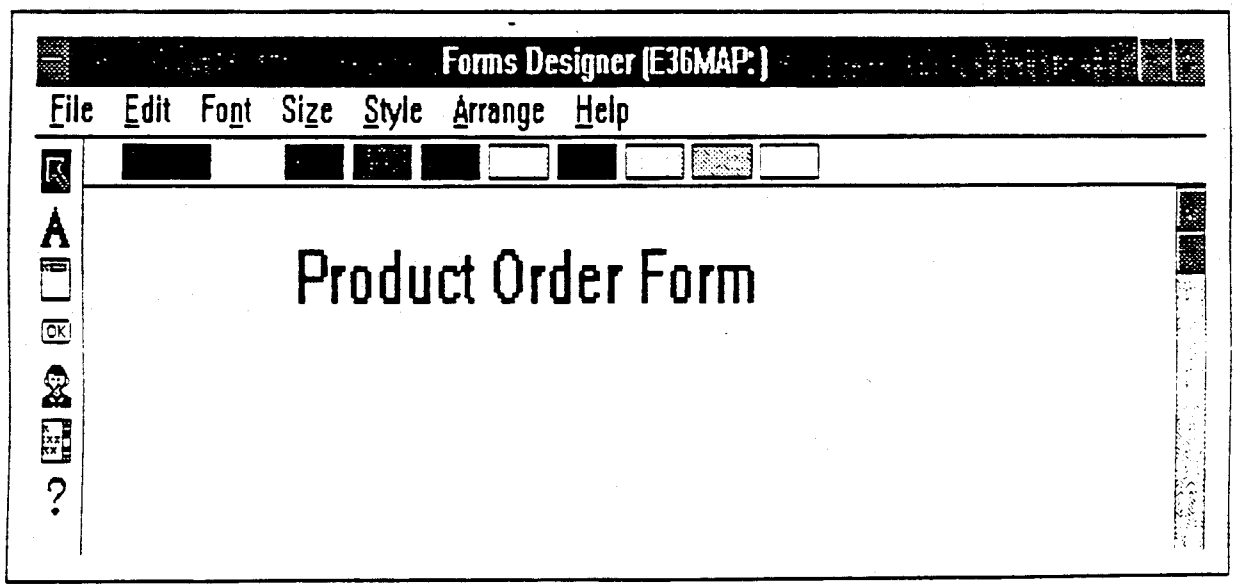
Visual 2: The text is redisplayed in the new size.

Sizing

- **Select 'Size', select text size**



- **Text is enlarged**



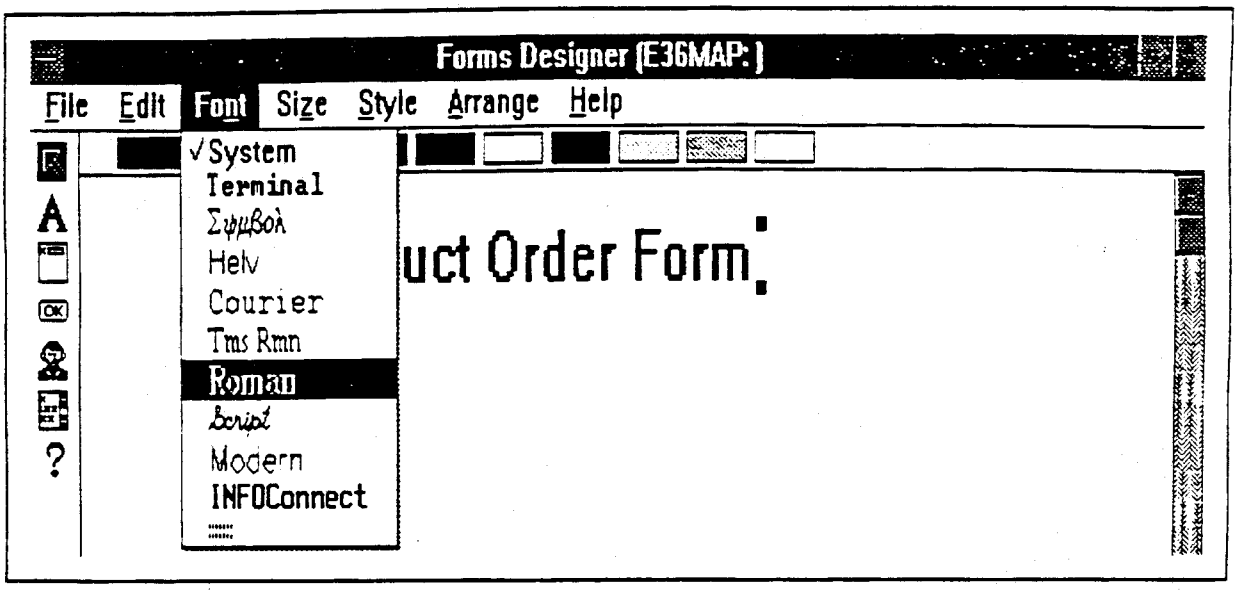
Changing Font

Visual 1: Text fonts can also be changed. Select the text to change, select 'Font', select font type.

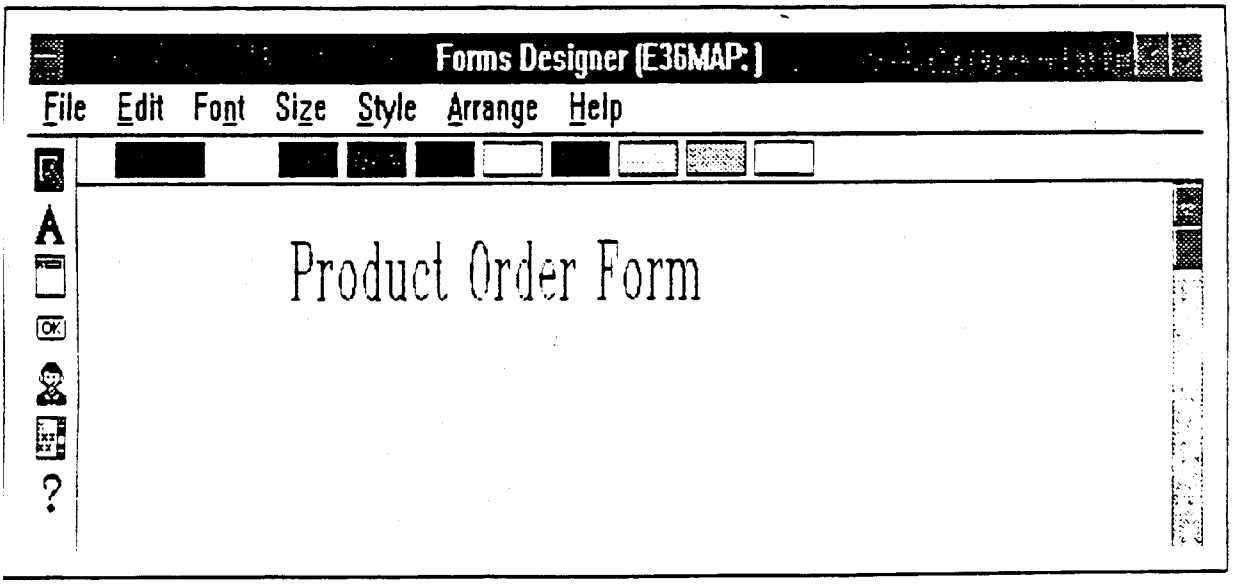
Visual 2: The text appears in Roman font.

Changing Font

- **Select 'Font'**



- **Text appears in new font**



Changing Text Style

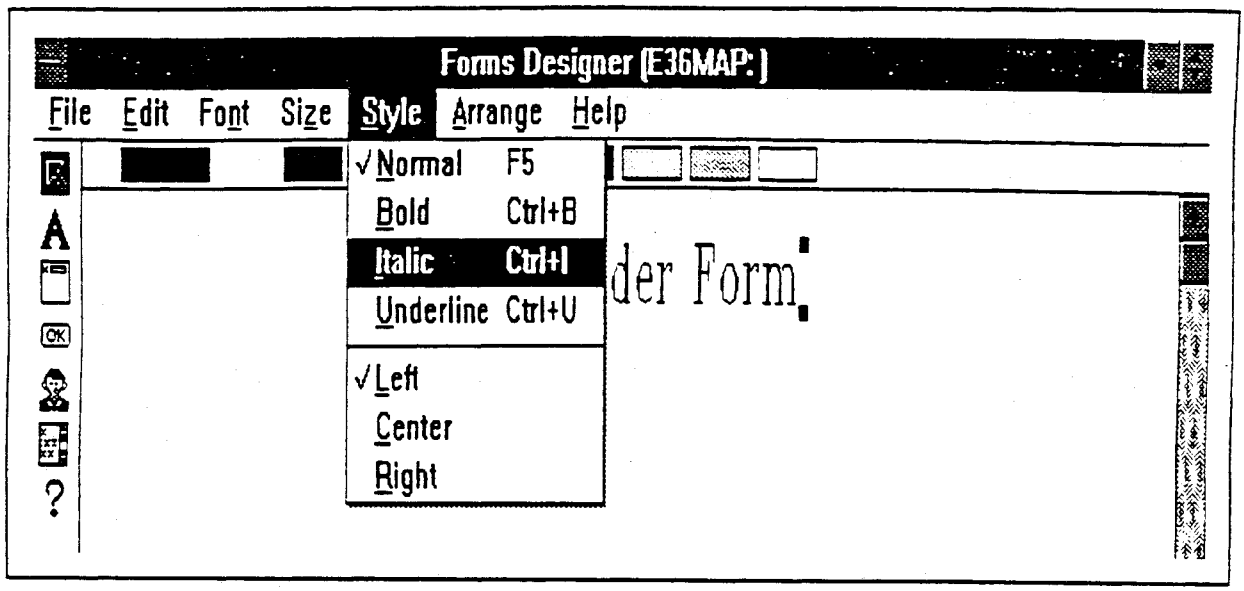
Visual 1: To change the style of the text, select text to edit, select 'Style', choose the style. In this example we wish to make Product Order Form italicized.

Visual 2: Text appears in new style.

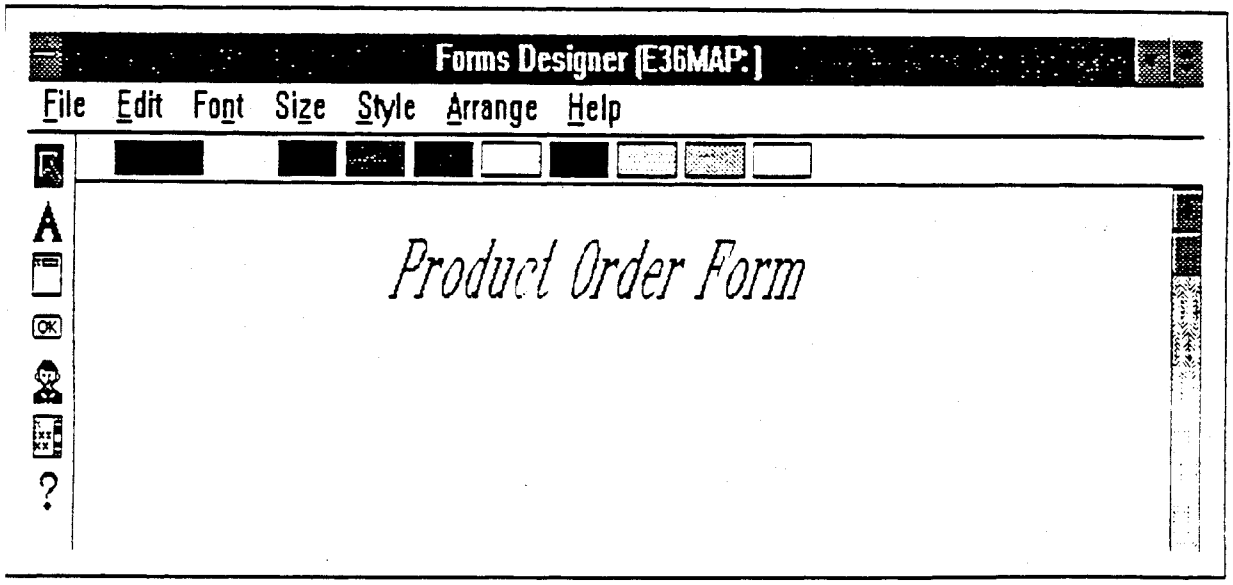
Note: *For more information on adding and editing text, refer to pages 3-9 through 3-11 of the DW Developer Training Guide.*

Changing Text Style

- **Select 'Style', select 'Italic'**



- **Text appears italicized**



Field Box Tool

Fields can be added and defined by using the Field Box tool. In our example, we wish to add the first of three input fields. The first field will solicit a product type to order. Text already has been added to the form after which this field will be placed.

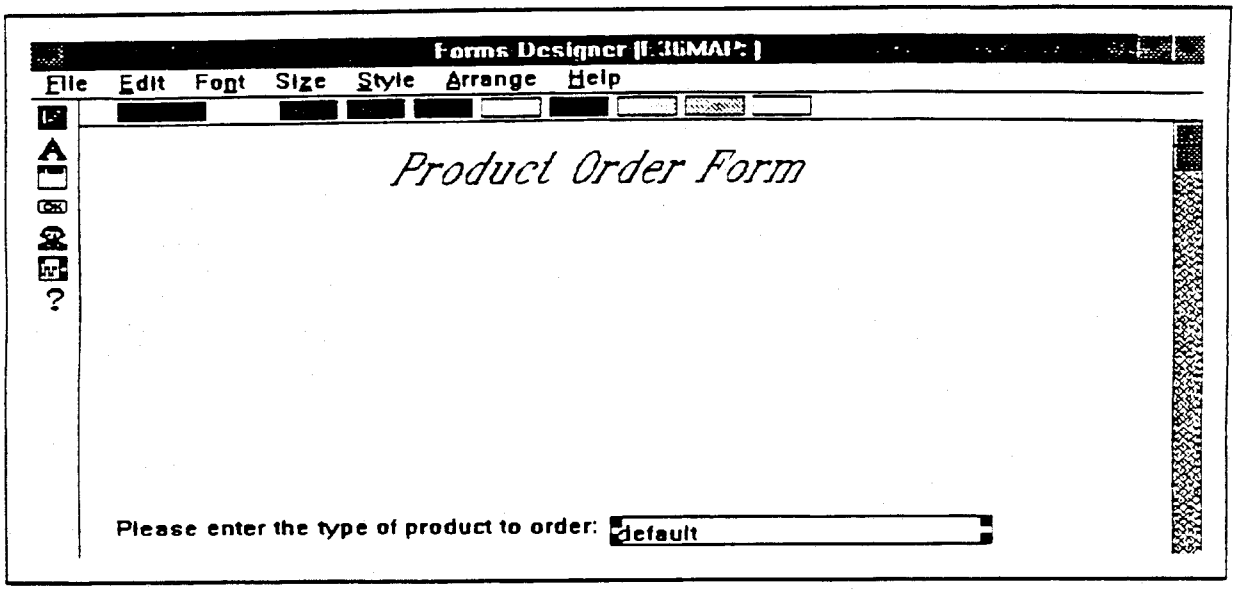
Visual 1: Begin by selecting the Field tool icon from the left side of the Forms Designer window. Use the mouse to define the field. Once the box is drawn, double-click on the box to produce the Field Characteristics screen.

Visual 2: The Field Characteristics screen is displayed. In our example, we will use the default settings. Click on the OK button to continue.

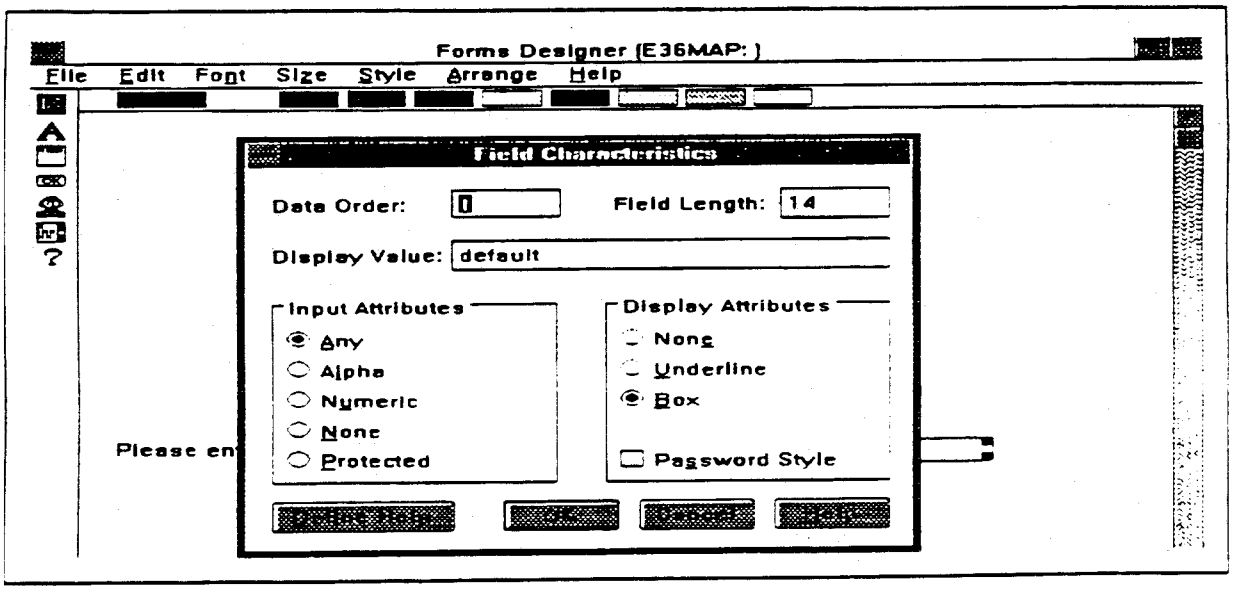
Note: *For more information on the field tool, refer to pages 3-12 through 3-15 of the DW Developer Training Guide.*

Field Box Tool

- Select field box tool icon



- Field Characteristics menu



Define Help

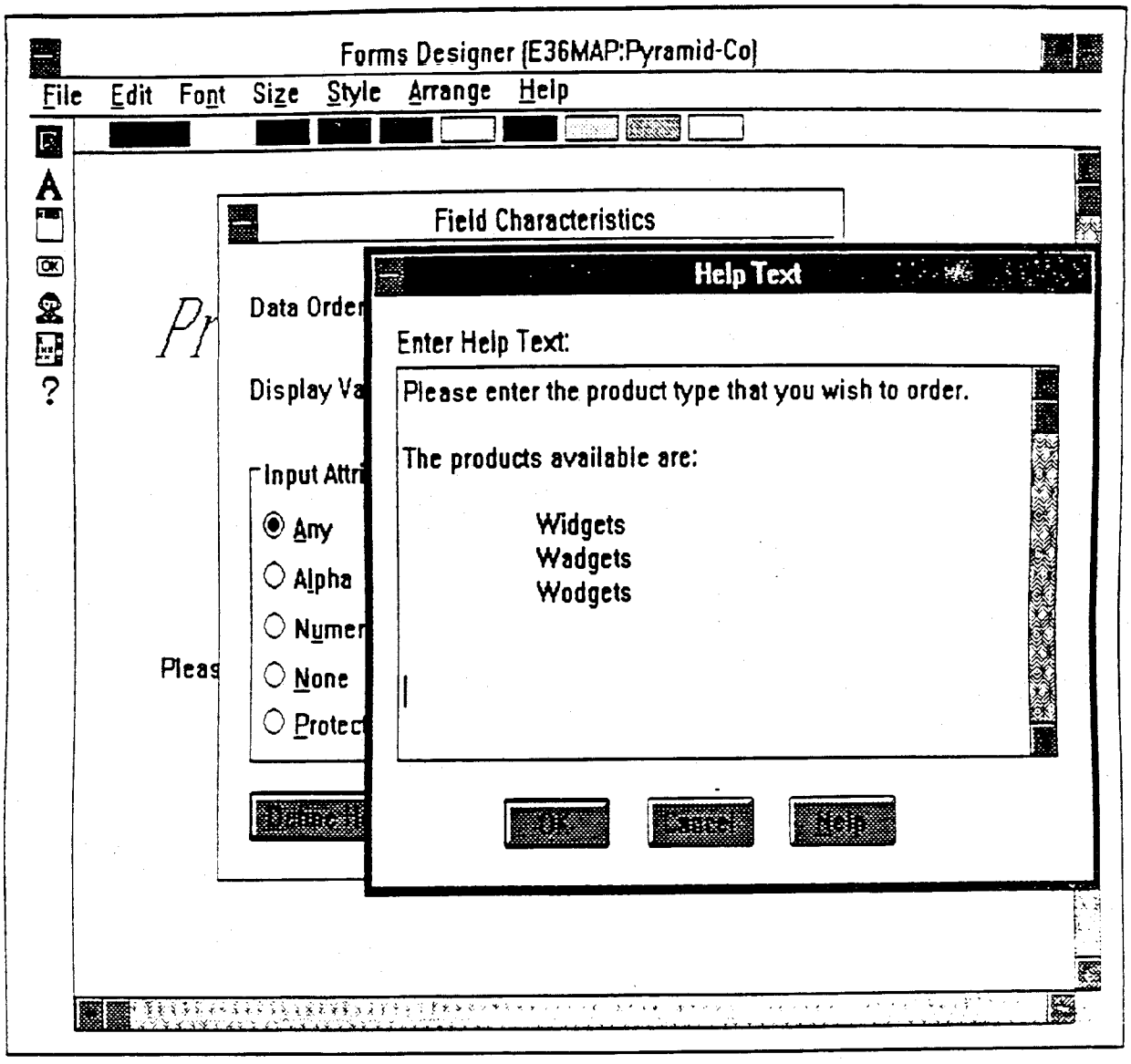
Several of the tool functions allow context help screen definitions. For example, a form may display an input field. The user may not be sure of what kind of information is acceptable. By defining context help for the field, the user is enabled to position the cursor in the field, press F1, and receive customized, field-specific help.

Context help can only be tested when an application calls the form.

Visual: To define help, click on the highlighted 'Define Help' button at the bottom of the object characteristic definition screens. In this example, we are adding help for a field in our form. The help text is entered. Select OK to save.

Define Help

- **Help Text screen**



Button Tool

We will now add a set of buttons to the form. The button group will consist of two selections, OK and Cancel. There are three different types of buttons to choose from:

- Command buttons
- Check boxes
- Options buttons

In this example we will create Command buttons.

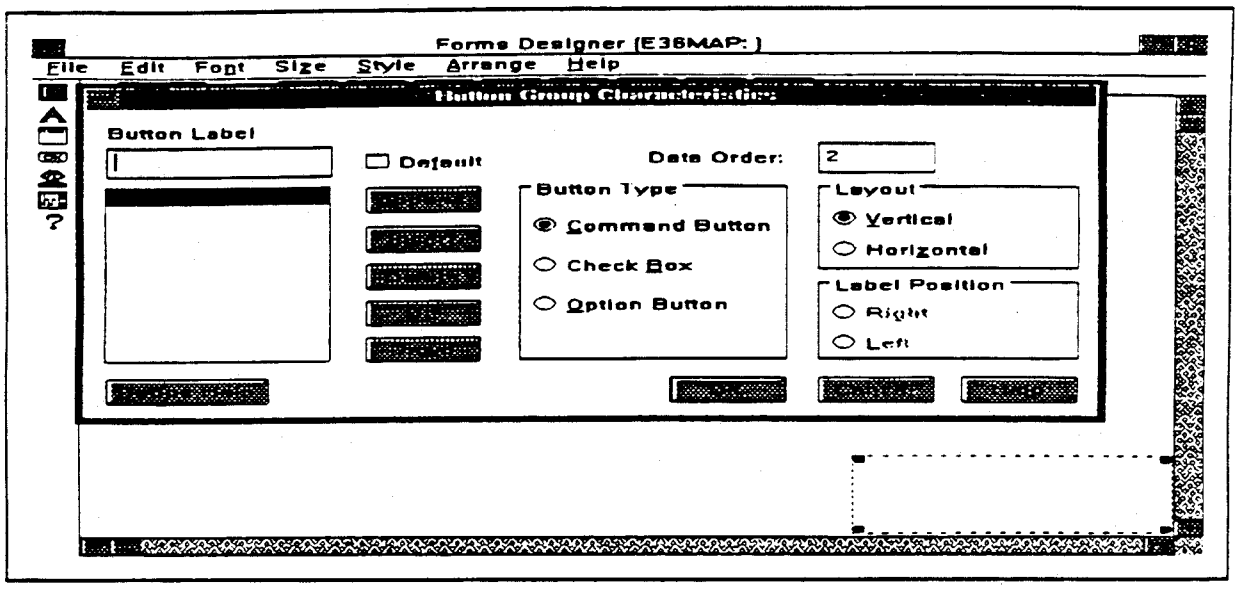
Visual 1: Select the button tool icon, position the cursor on the form work-screen, and create a box. Double-click on this box to produce the Button Group Characteristics screen.

Visual 2: Add the button labels one at a time in the Button Label field and select Append, Insert, Delete, Up, and Down buttons to arrange the labels added.

Note: *For more information on using the button tool, refer to pages 3-16 through 3-19 of the DW Developer Training Guide.*

Button Tool

- **Select button tool icon, Button Group Characteristics menu**



- **Fill in Button Group Characteristics menu**

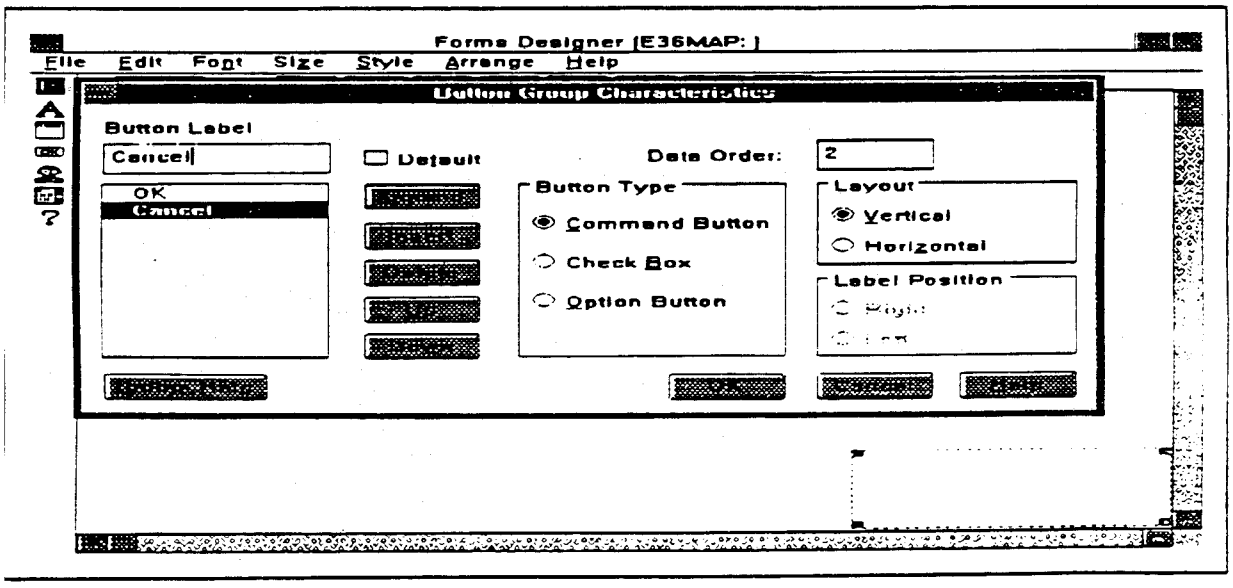


Image Tool

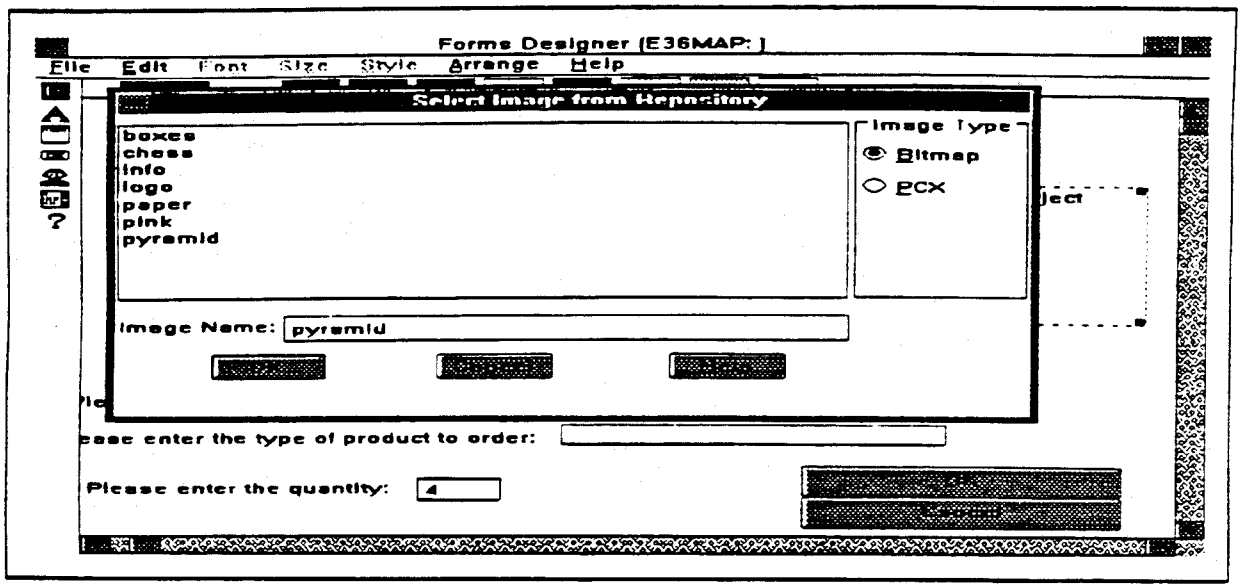
Images can be added to the form. These images can be graphics, pictures, etc. Any file in BMP or PCX format can be used. These files must exist in the current working Repository Partition. If the image objects do not currently exist in the partition, they will have to be imported. In this example, we will add an image file to the form. The image object is a BMP file called Pyramid.

Visual 1: Select the image tool, draw a box for the image, double-click on the box. The Select Image from Repository window is displayed. Select the object to be loaded into the image box.

Visual 2: The image appears on the form work-screen.

Image Tool

- Select image icon, choose 'pyramid' image from repository list



- Form with 'pyramid' image added

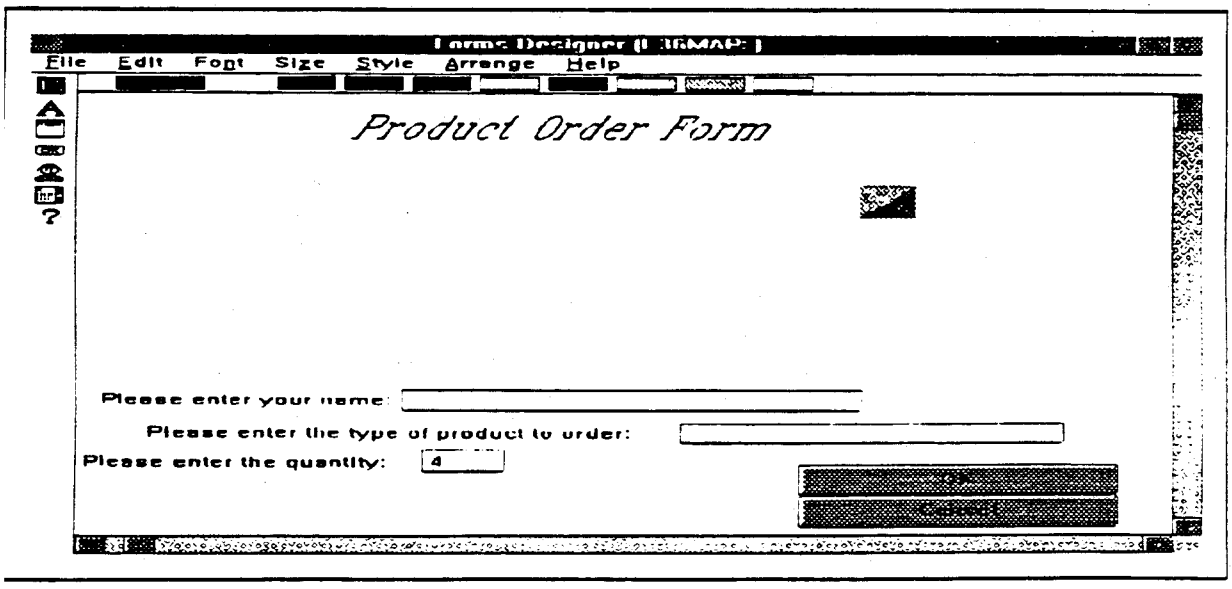


Image Tool

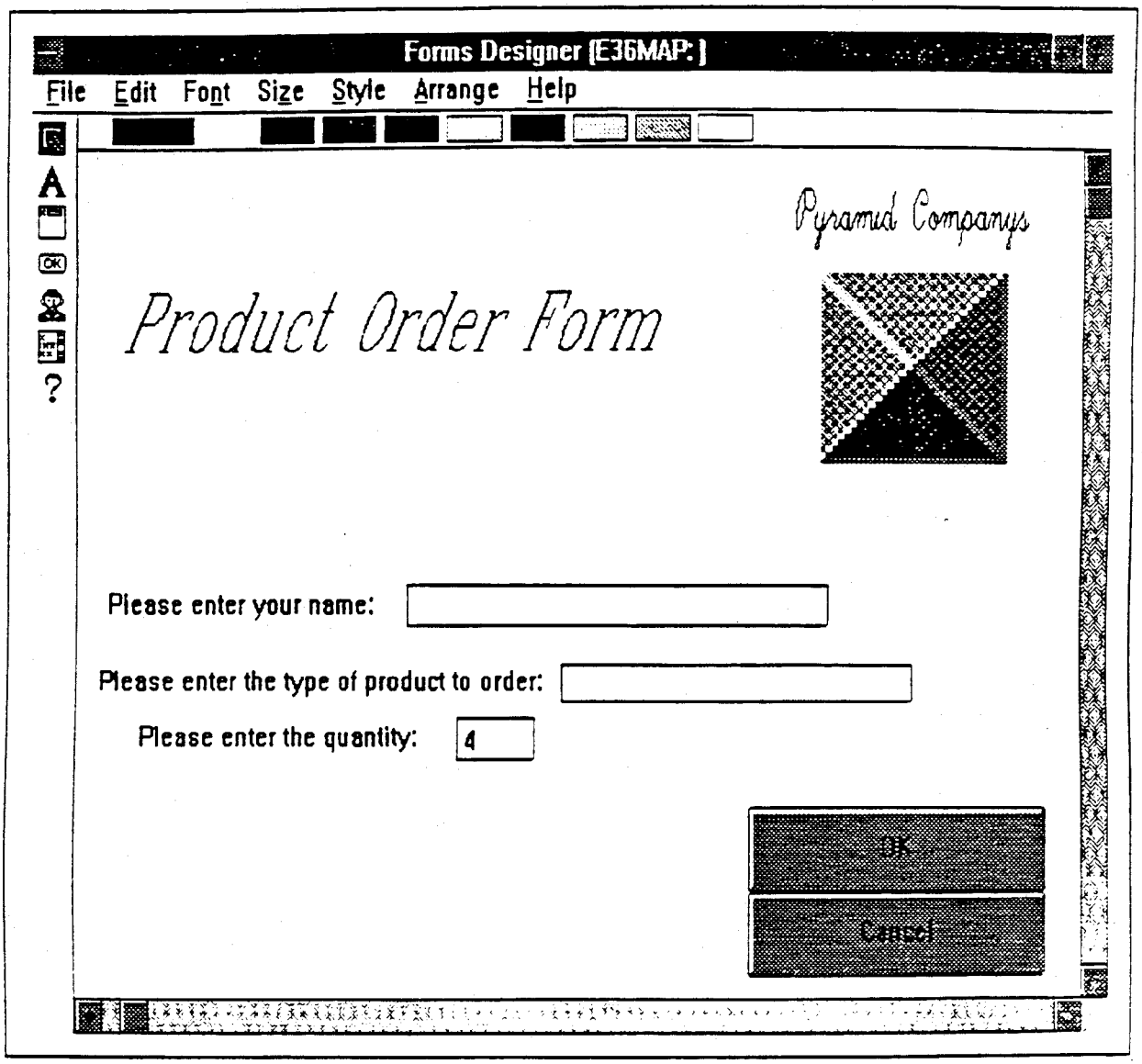
The size of the images brought into the Forms Designer work-screen will not match the size of the box created with the mouse. The image retains it's initial size before it is brought into the Forms Designer. However, once the image is displayed in the Forms Designer work-screen, it is possible to click on the object, and 'size' the image.

Visual: In our example, the image was enlarged and text was added above it.

Note: *For more information on the image tool, refer to pages 3- 20 through 3-22 of the DW Developer Training Guide.*

Images

- **Form with enlarged image with added text**



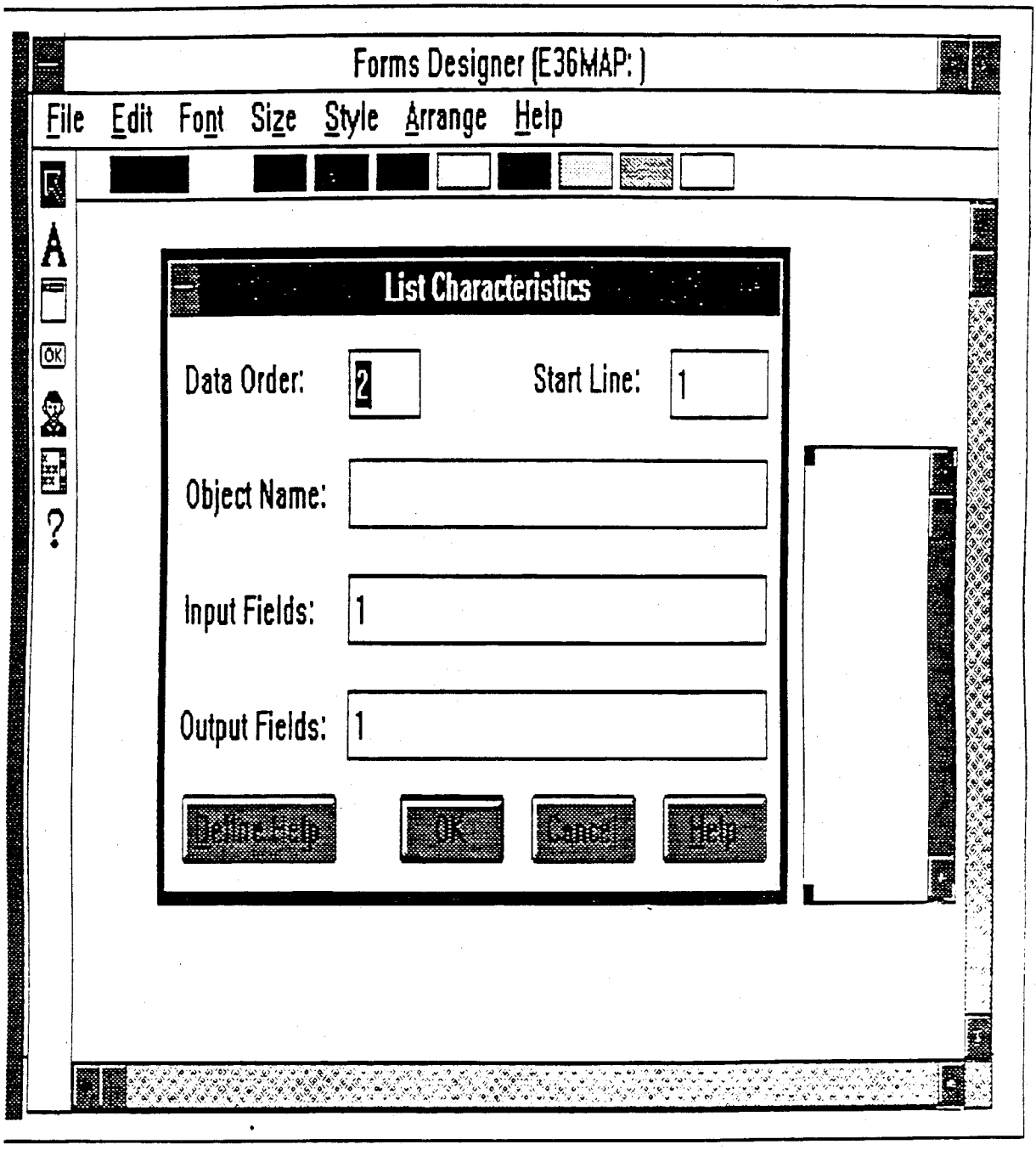
List Box Tool

Note: *We will not add a list box to our form example. The List Characteristics menu is displayed on the opposite page. For information on creating and using list boxes, refer to:*

Adding a List Box: *DW Developer Training Guide pages 3-23 through 3-26.*
List boxes and Linc: *DW Developer Training Guide pages 7-7 through 7-8.*
Use with MAPPER Application: *DW Developer Training Guide Sections 4,5,6.*

List Box Tool

- **List Characteristics screen**



Help Tool

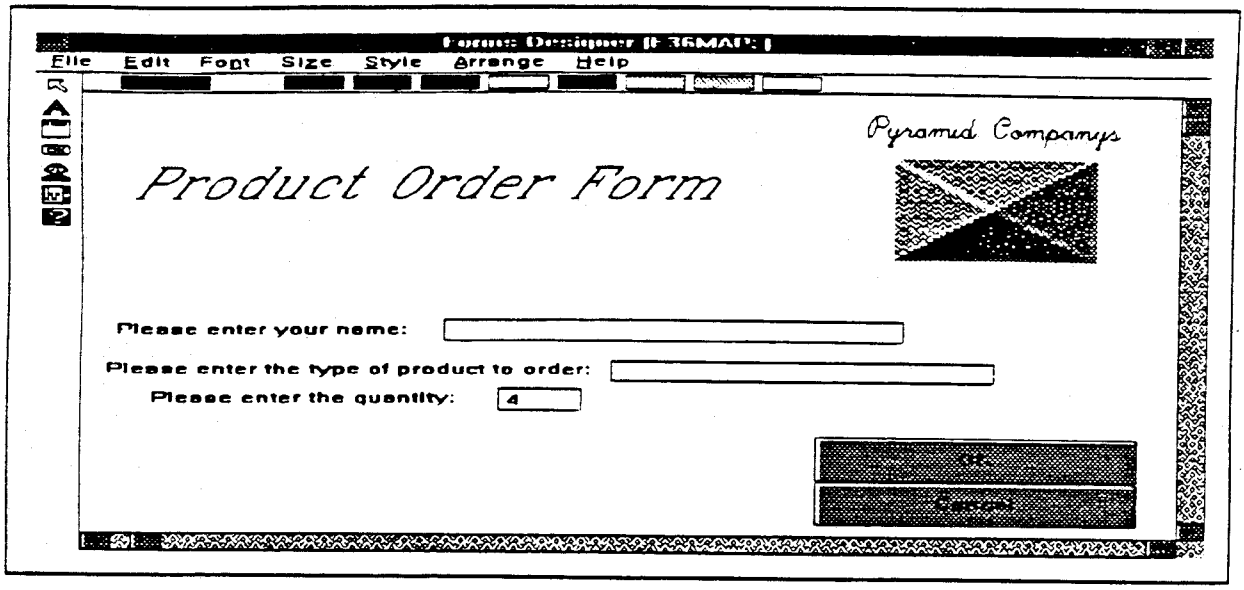
There are three ways to obtain help when working with the Forms Designer. We have already seen how help can be obtained by selecting buttons on object characteristic screens. It is also possible to obtain help from the Forms Designer menu bar. Finally, help can be obtained by selecting the ? icon from the tool list. By selecting the ? icon you can then click on any of the other tool icons to receive 'Tools' help.

Visual 1: Select the ? icon and click on one of the other tool icons. In this example we will click on the List tool icon.

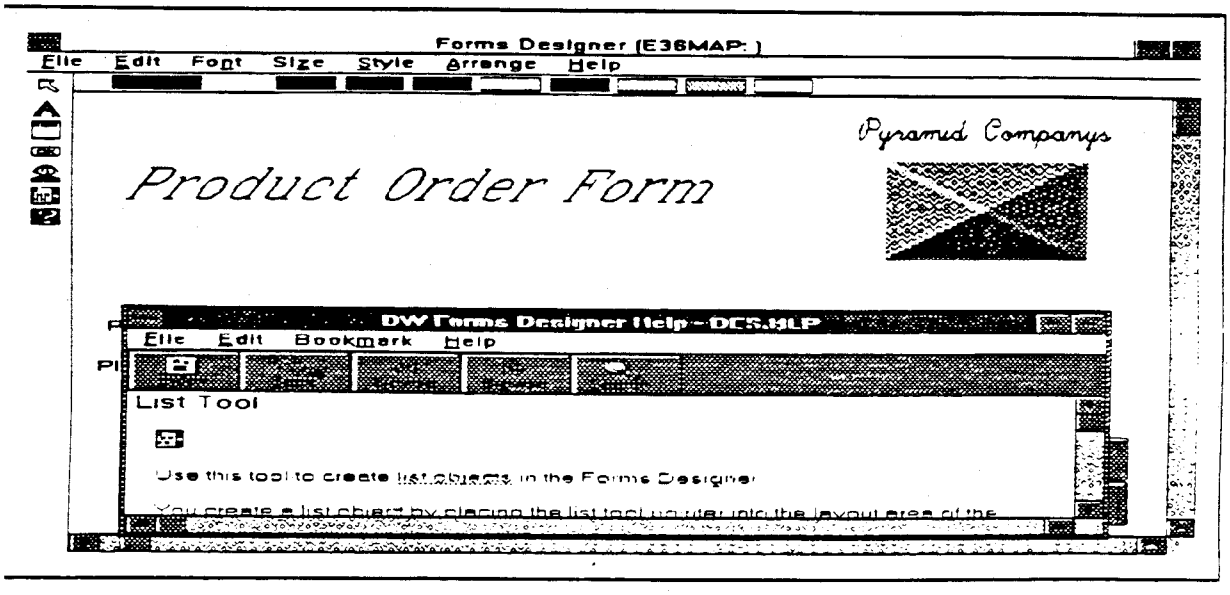
Visual 2: A help window for the List box icon appears.

Help Tool

- **? icon selected, request help for list box tool**



- **Help window appears**



Color Options

Earlier we looked at how to edit and change text. We can also select colors for the text in a form. There are two ways to use color:

- Color palette
- Custom color option

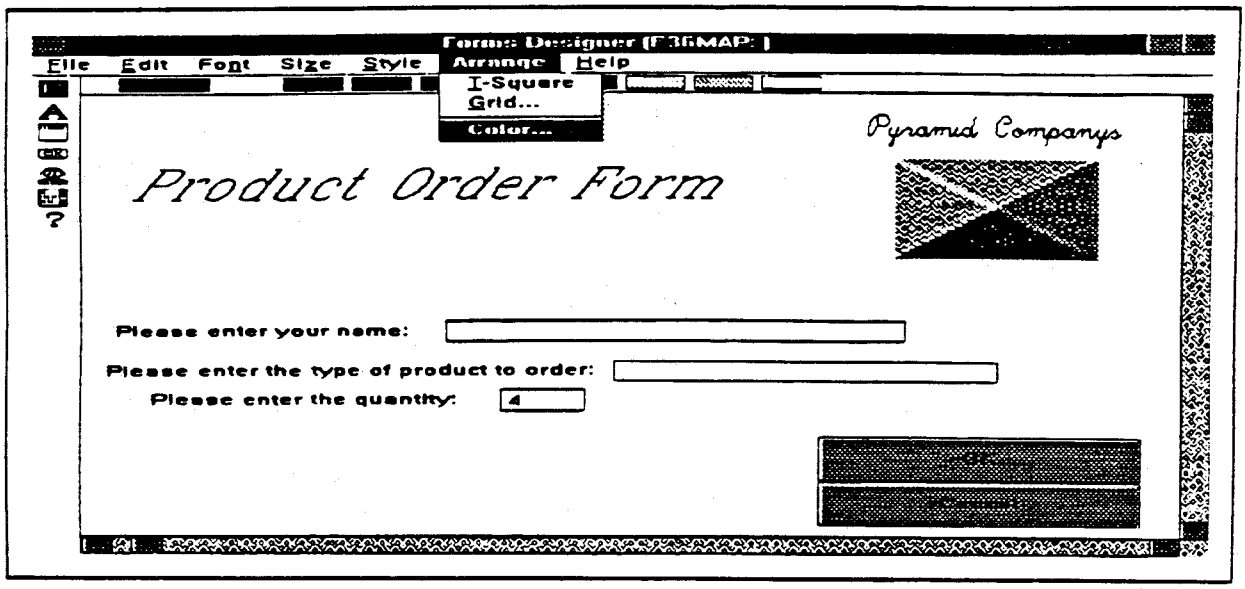
Use the mouse to click on a section of text to color. Then use the mouse to select a color from the color palette located below the menu bar at the top of the Forms Designer window.

Visual 1: Colors can also be customized. Select 'Arrange' from the menu bar, select 'Color'.

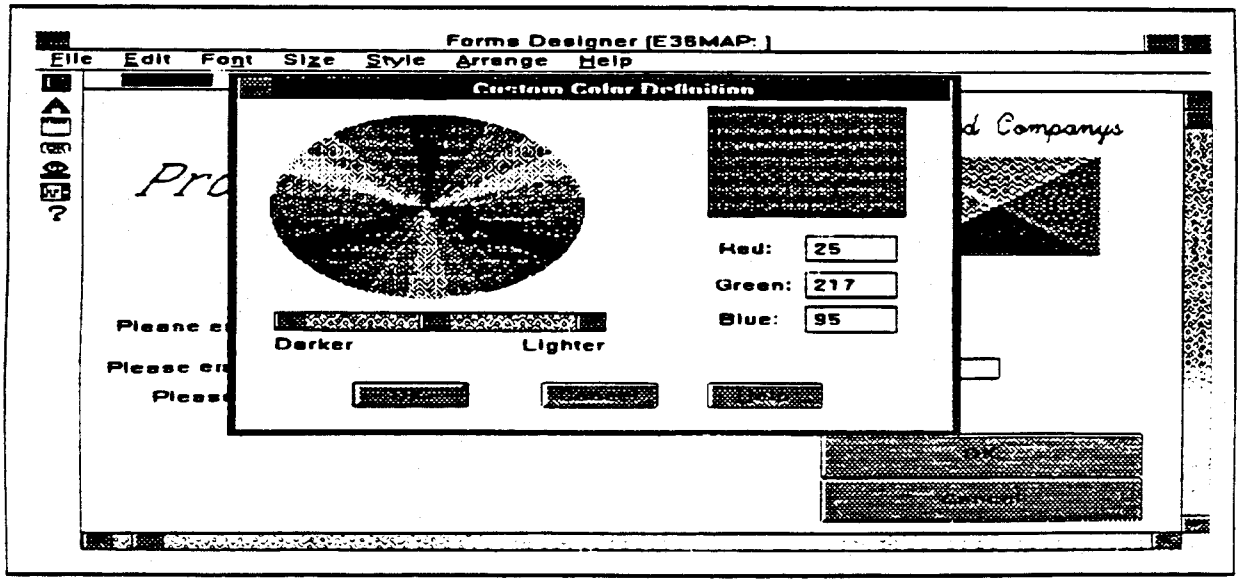
Visual 2: The Custom Color Definitions screen is displayed. Use the Darker/Lighter slide to determine the brightness of the colors. Then use the mouse to click on the color of choice.

Color Options

- **Select 'Arrange', select 'Color'**



- **Custom Color Definition screen**



Saving the Form

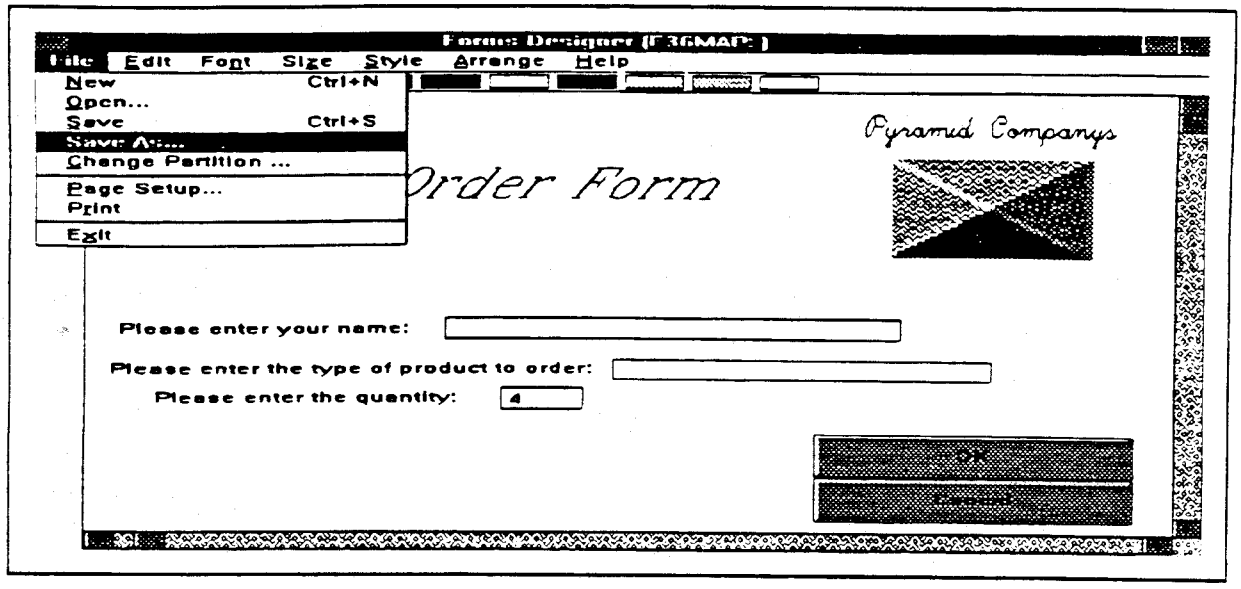
Forms can be saved by using features found in the File menu bar list. Use 'Save As' to save a new form. Use 'Save' to replace the existing copy.

Visual 1: In this example we will save our form for the first time. Select 'File', select 'Save As...'

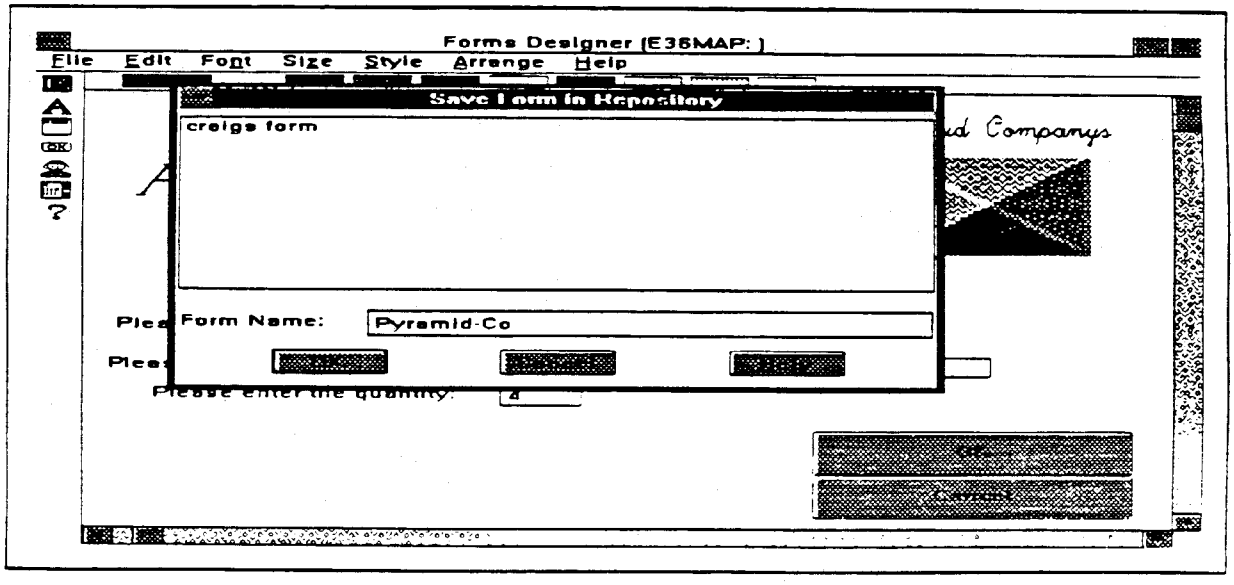
Visual 2: The Save Form in Repository screen is displayed. Enter the name for the new form in the Form Name field. We will call this form 'Pyramid-Co'.

Saving the Form

- Select 'file', select 'Save As'



- Save screen



T Square

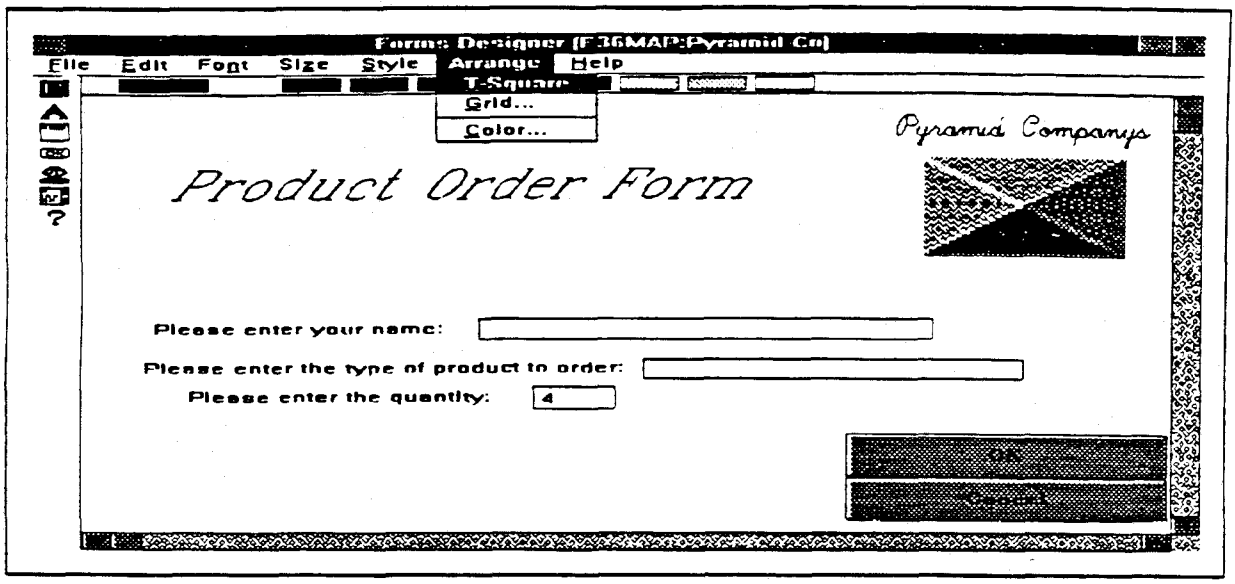
Use the T-Square feature to 'line-up' and arrange form objects.

Visual 1: Select 'Arrange', select 'T-Square'

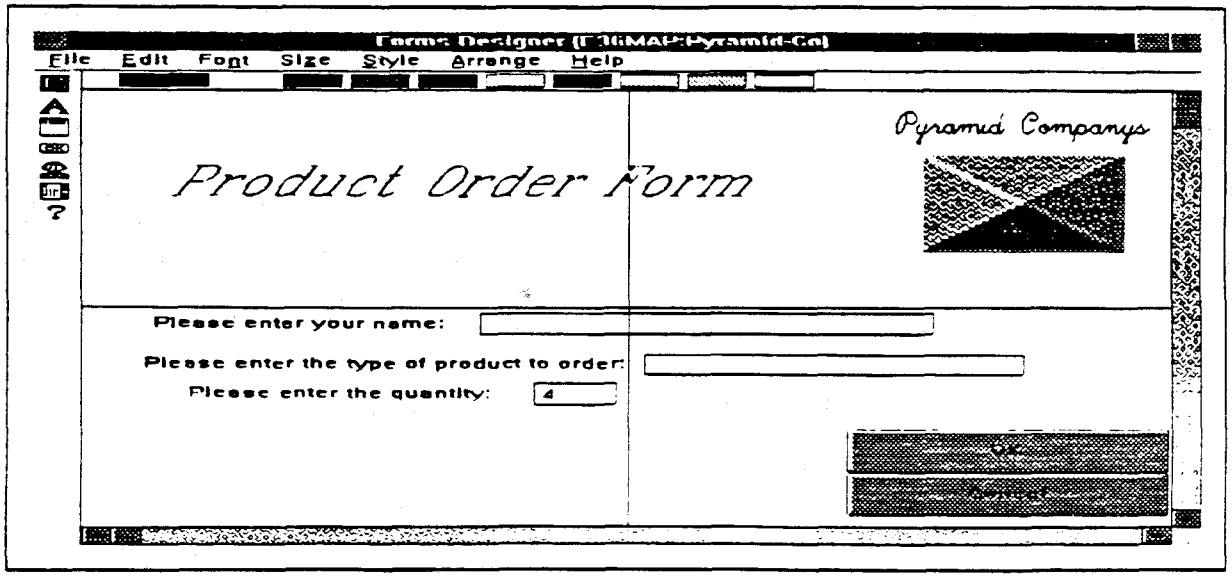
Visual 2: T-Square is displayed on the form. To remove, select T-Square and Arrange from the menu bar.

T-Square

- **Select 'Arrange', select 'T-Square'**



- **Form with T-Square**



Grid

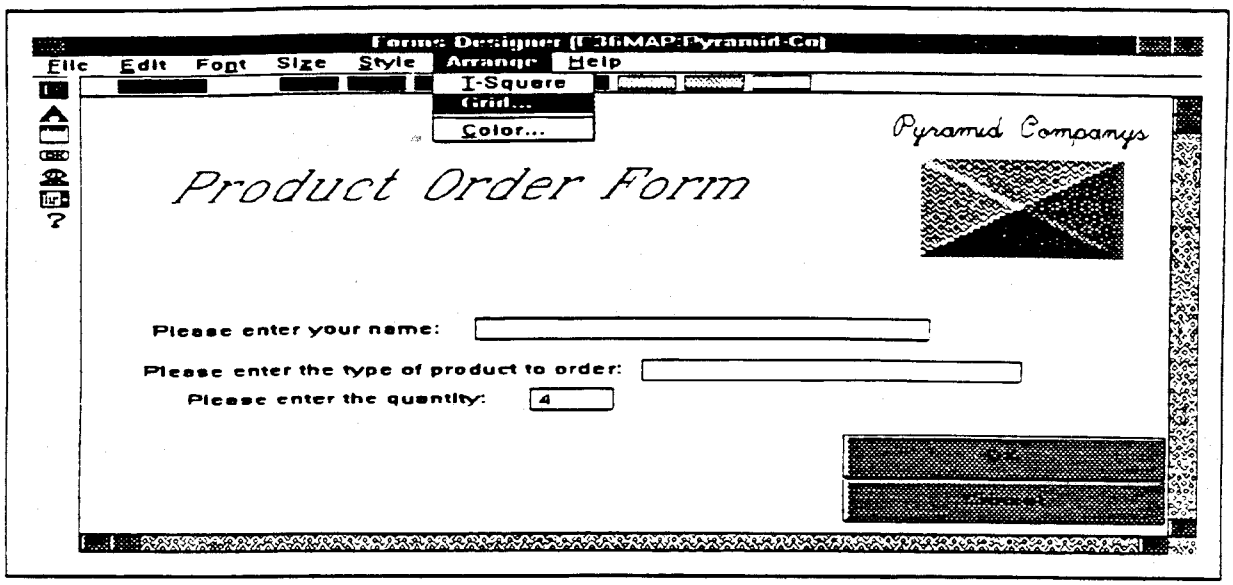
Grid is another tool used for arranging form objects. In this example, we wish to 'line-up' the three text objects for the input fields.

Visual 1: Select 'Arrange', select 'Grid'.

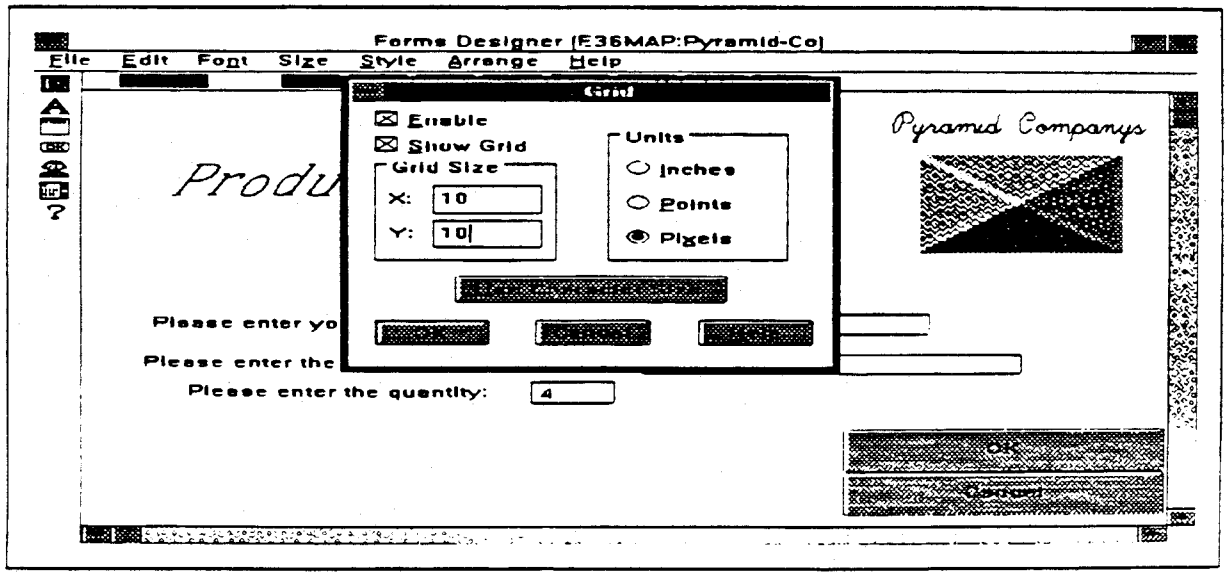
Visual 2: Grid menu is displayed. Here you determine the characteristics of the grid to display. Select OK to continue.

Grid

- **Select 'Arrange', select 'Grid'**



- **Grid menu**



Grid

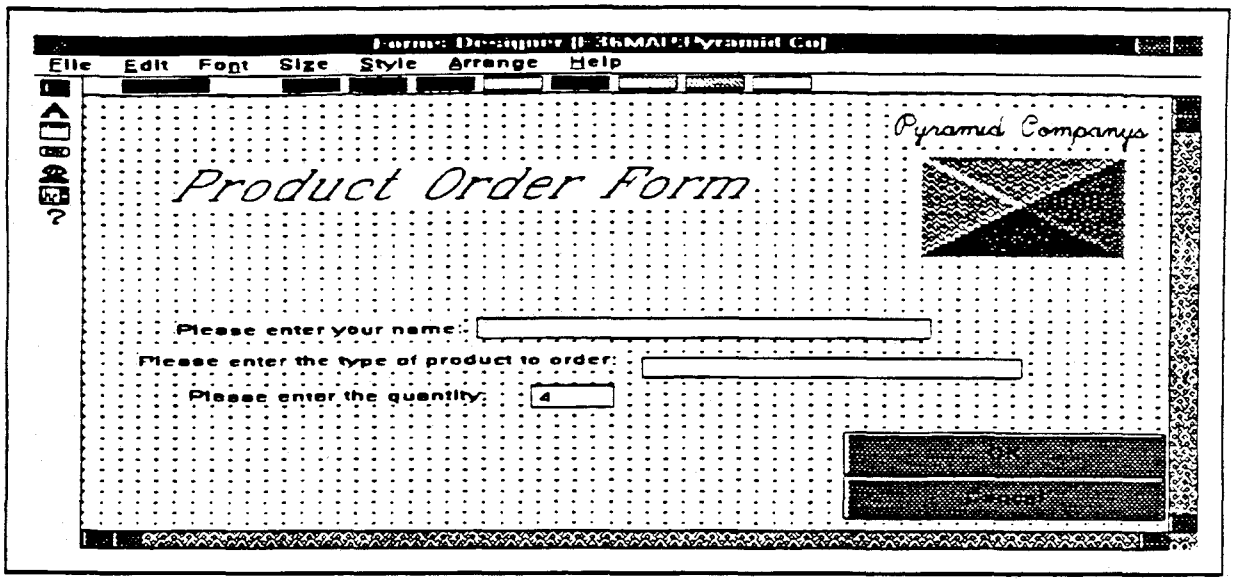
Visual 1: The grid is displayed. Use the mouse to select and move the text groups. The grid will force the groups into specific location positions.

Visual 2: Form with text properly arranged.

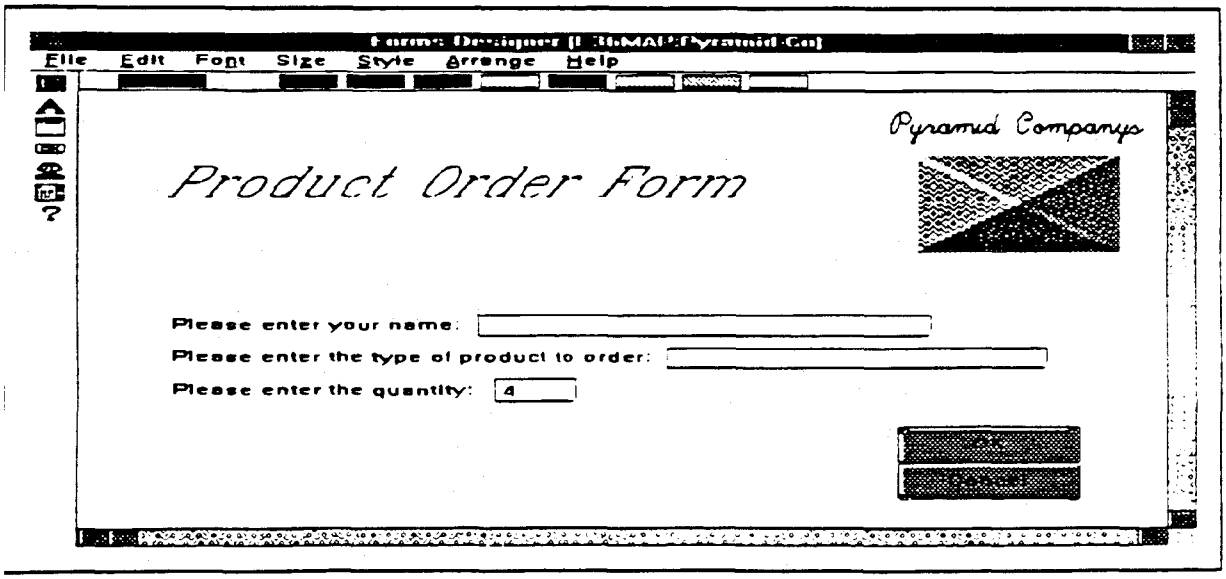
Note: *For more information on T-Square and Grid, refer to pages 3-29 through 3-32 of the DW Developer Training Guide.*

Grid

- **Form with grid displayed**



- **Field text is now lined up**



Exercise

1. Create an input form utilizing various object tools and options.

Exercise 1: Creating the Sample Input Form

During this exercise, you create the sample input form used to select office supplies.

Input form contents

The input form for the sample application contains the following items:

- Text.

  The title of the form and labels for fields and boxes.

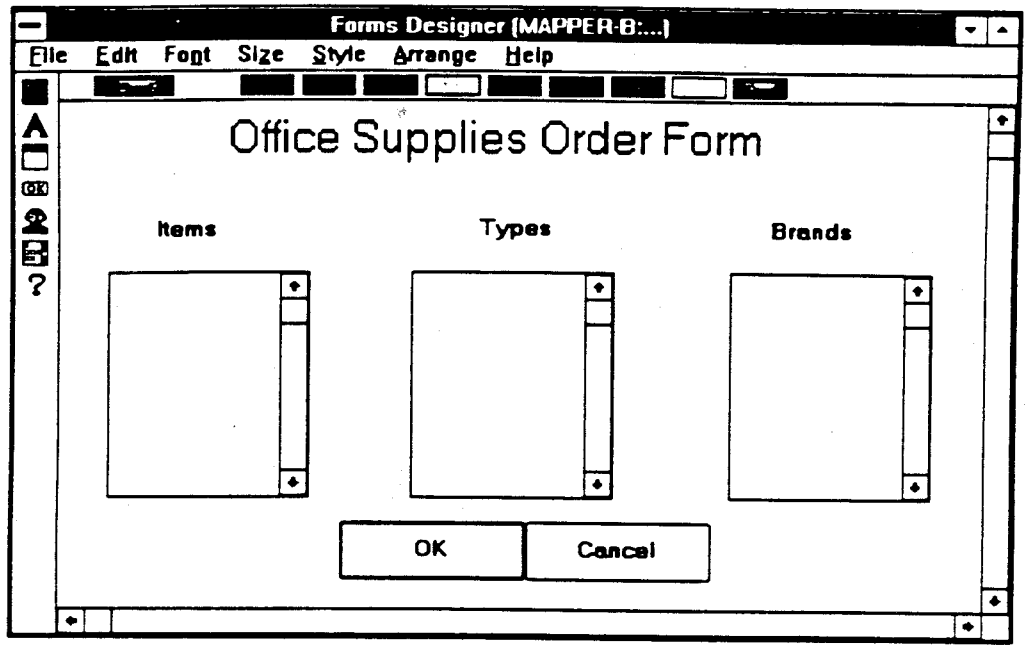
- Three list boxes.

  One list box for office supply items (erasers, pens, and so on), one for types of items (such as marker pens, ballpoint pens, and so forth), and one for brands (such as E-Z Mark).

- One button group with two command buttons.

  The command buttons allow the user to display the inventory (OK) or to cancel the request.

Sample input form



Overview of input form creation

The following sections provide detailed instructions on how to perform each of these steps:

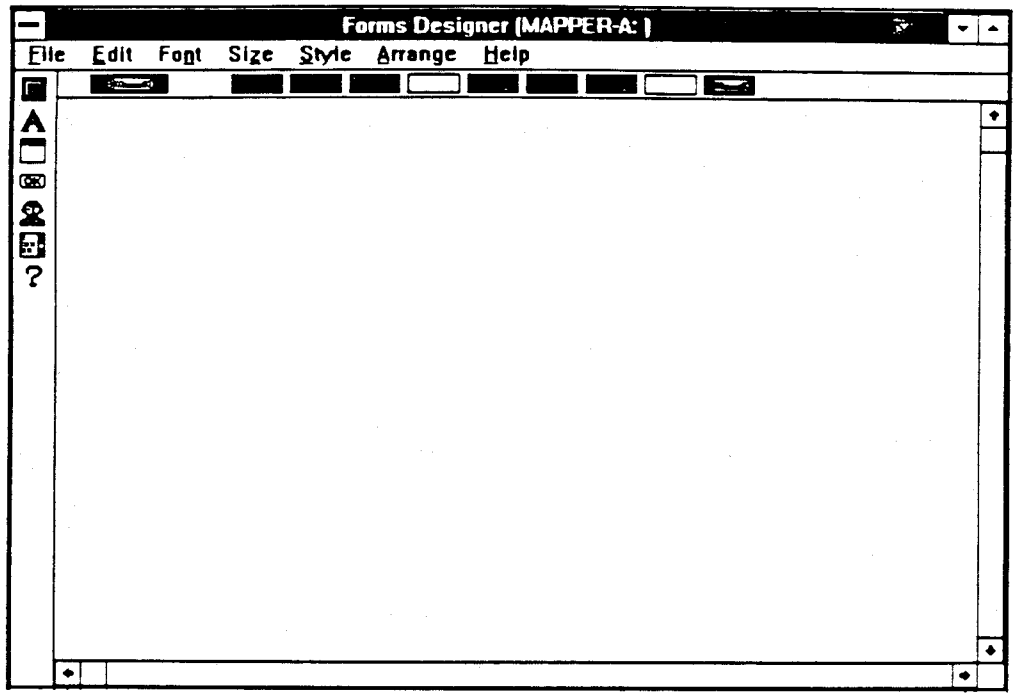
1. Start the Forms Designer.
2. Add text to the blank form.
3. Create the list boxes for the form.
4. Create the button group for the form.
5. Edit the form (if necessary).
6. Save the form in the Repository.

Starting the Forms Designer

To start the Forms Designer, double-click the Forms Designer icon in either the Designer Workbench group window or main window. You may have to sign on to Designer Workbench and choose a partition.

The Forms Designer screen appears. (See Section 3 for more information on starting the Forms Designer.)

Note: *Before you start editing your form, you may want to enable the grid in order to accurately place objects and text. (See Section 3 for more information on using the grid.)*

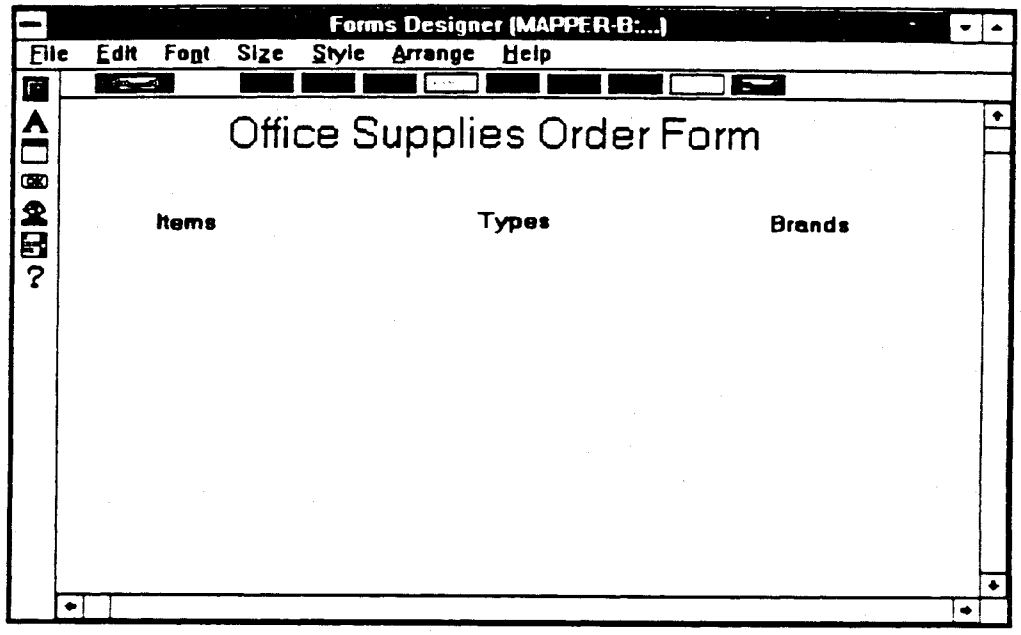


Entering Text for the Input Form

Use the following procedure to enter text for the form:

1. Click on the text tool.
2. Move the pointer to where you want to enter the text and click.
3. Type **Office Supplies Order Form**
4. Repeat steps 1-3 for three text fields, typing **Items**, **Types**, and **Brands**.

   The result should look like the following screen.



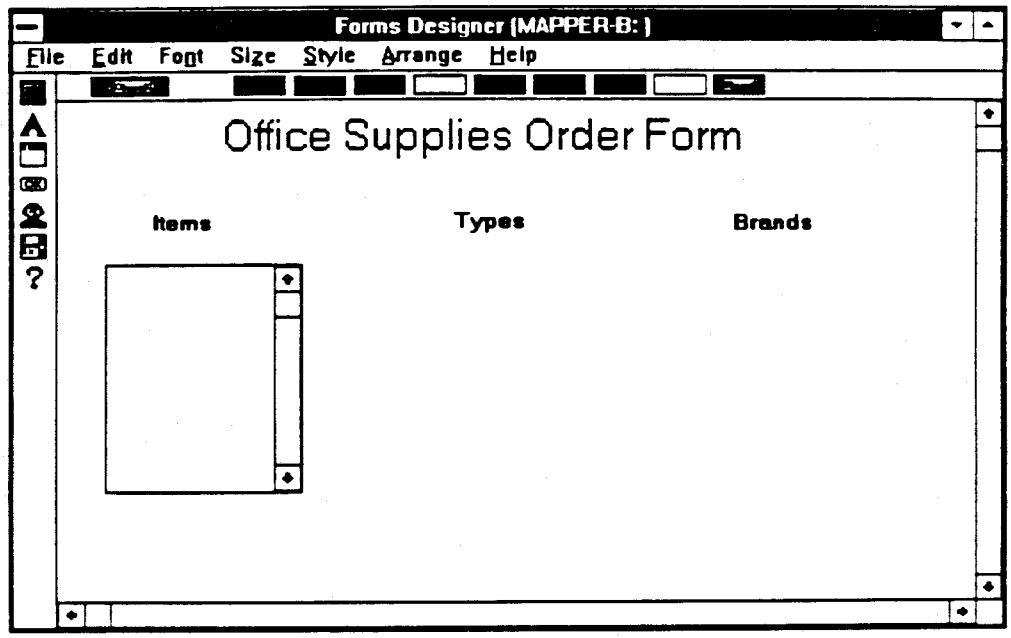
Adding List Boxes to the Input Form

The input form has three list boxes: one to display office supply items, one to display types of items, and one to display brands.

Use the following procedure to add the list boxes for the sample input form:

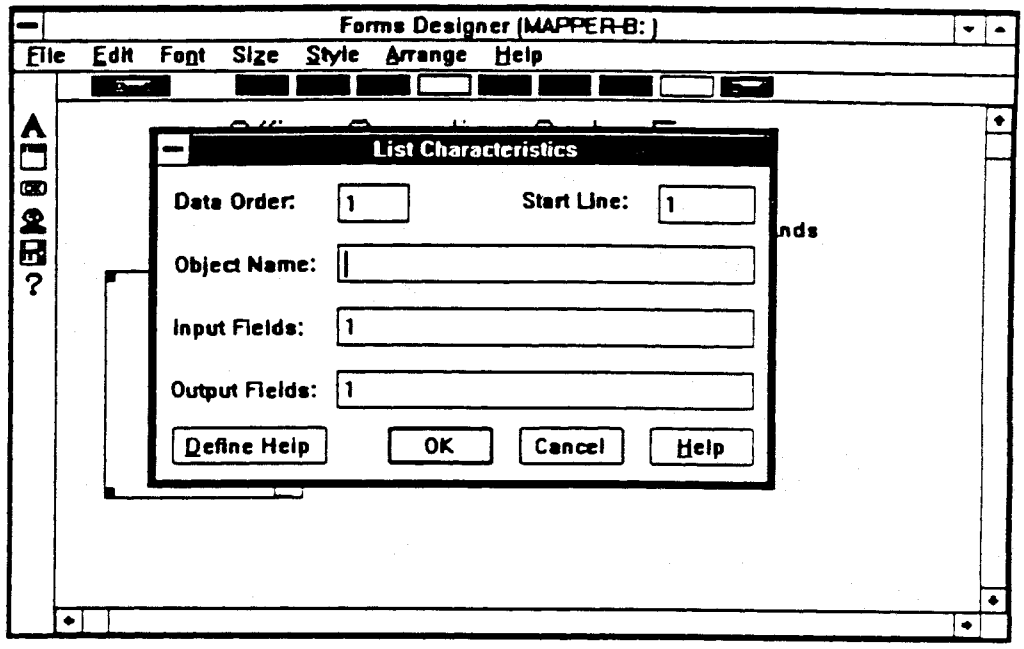
1. Click on the list box tool.
2. Move the pointer to where you want the upper left hand corner of the list box.
3. Click and drag the mouse to correctly size the list box.
4. Release the the mouse button.

   The result should look like the following screen.



5. Double-click the list box object.

   The List Characteristics dialog box appears.

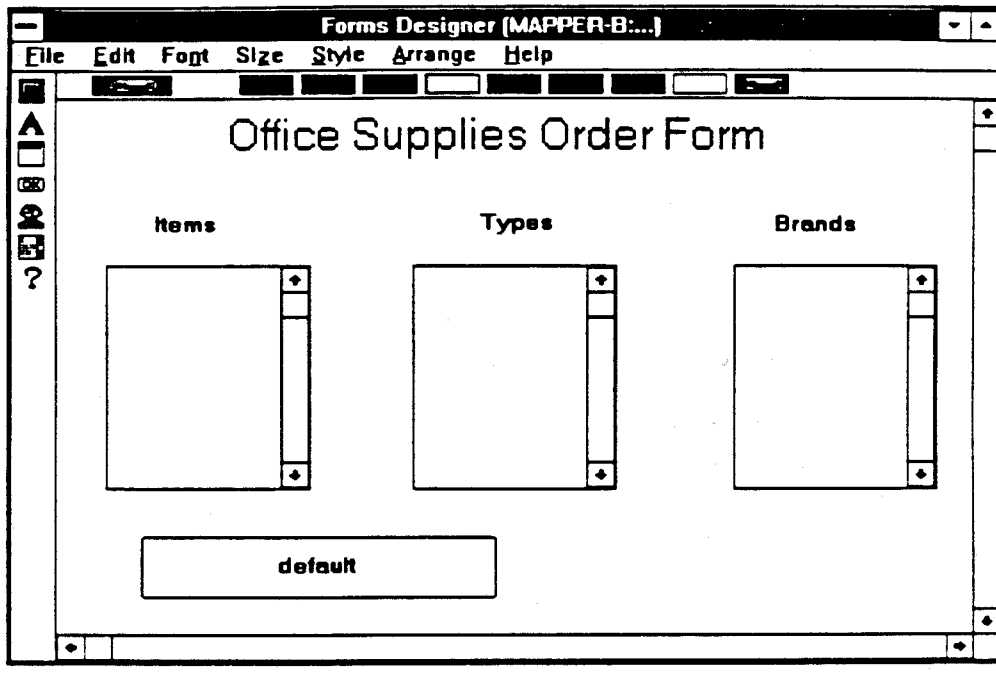


6. Fill in the first dialog box to match the one shown in the screen above. (The Object Name field is not filled in.)
7. Choose OK.
8. Repeat steps 2-7 to add a list box for Types and one for Brands, but use data order 2 for the Types list box and data order 3 for the Brands list box.

Adding the Button Group to the Input Form

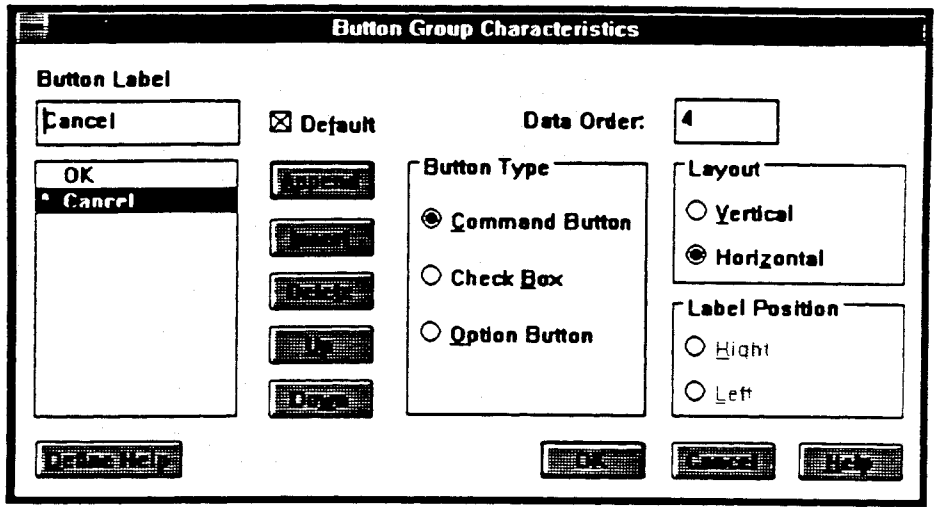
Use the following procedure to add the button group for the sample input form:

1. Click on the button tool.
2. Move the pointer to where you want the upper left hand corner of the button group.
3. Click and drag the mouse to correctly size the button group.
4. Release the mouse button. The result should look like the following screen.



5. Double-click the button group.

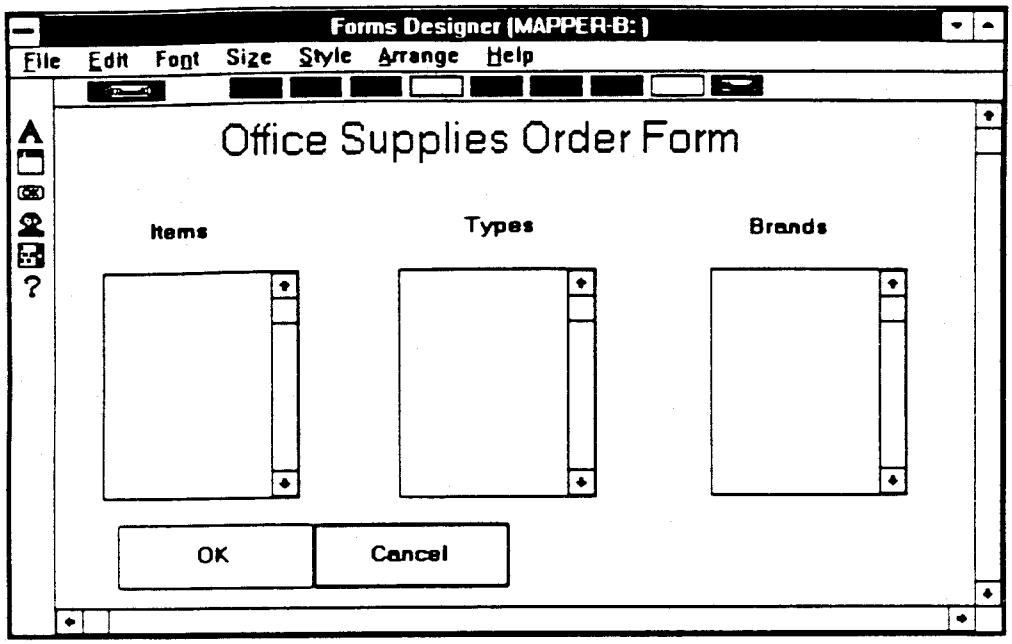
The Button Group Characteristics dialog box appears.



6. Fill in the Button Group Characteristics dialog box to match the one in the screen above.

7. Choose OK.

   The result should look like the following screen.



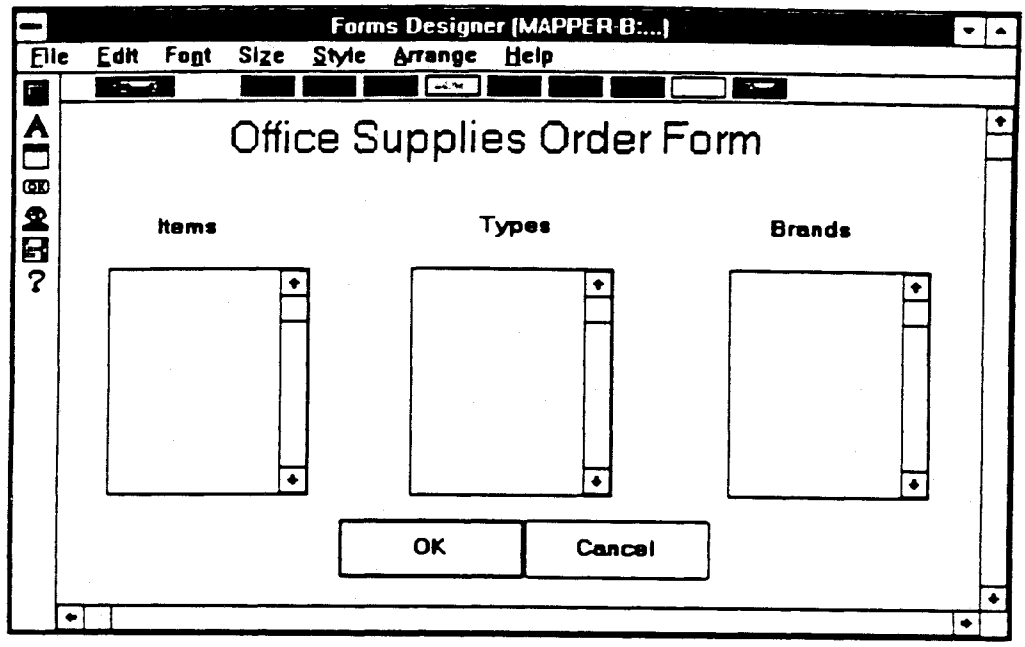
Editing the Sample Input Form

You can edit the sample form using the pointer tool. For example, use the pointer tool to center the OK-Cancel button group on the bottom of the form or change the size or style of the text. For instructions on editing forms, see Section 3.

Note: *Remember that proportional fonts do not necessarily appear the same onscreen at run time as they do at design time. Also, if text is displayed on a workstation that does not have the same fonts or screen resolution as the workstation the form was created on, the effects on the display of the form at runtime are unpredictable.*

Final result

The edited form should look like the following:

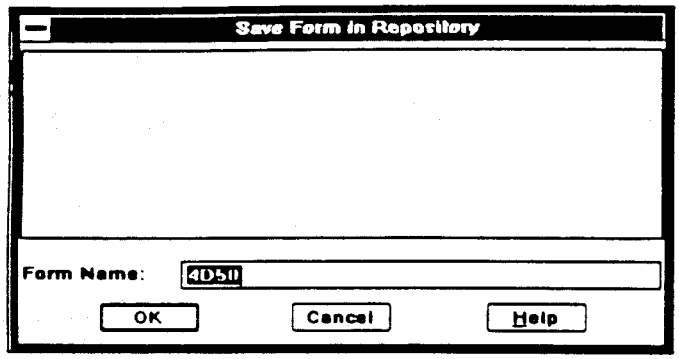


Saving the Sample Input Form

Now that you have added the input form, save it in the Repository. Use the following procedure to save the form:

1. Select the File menu.
2. Choose Save.

   The Save Form in Repository dialog box appears.



3. Choose OK to save the form under the filename in the Form Name field. If you choose a filename that already exists, the Repository verifies that you want to overwrite the file.

   Note: *If you save a form with a MAPPER report number as the filename, the filename must be in the normalized format of report, drawer (uppercase), cabinet (always an even number). For example, an object must be saved as 5D50, not 5d50 or 5d51.*

BTN (Define Button)

Use the Define Button (BTN) run statement to define a button and display it on the screen. Use this statement in conjunction with the Accept Input (INP) statement to enable the user to carry out an action with the button.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@BTN[,vwh,row,col,rsiz,csiz,fc/bc,o] text [vbh] .
```

The following table describes the fields and subfields:

Field	Description
vwh	Variable that contains an existing window or button handle. If you use a window handle in this field, the button is placed in that window and all row and column coordinates are relative to the specified window. Default = main window. If you specify a button handle, the system assumes you are changing that button in some way. If you do not specify a value in the *row*, *col*, *rsiz*, *csiz*, *fc/bc*, *o*, or *text* fields, the system uses the original values.
row	Row number of the upper left corner of the button. Default = 1.
col	Column number of the upper left corner of the button. Default = 1.
rsiz	Vertical size of the button, in rows. Default = 4. This size is based on the current font of the main window.
csiz	Horizontal size of the button, in columns. Default = 7. This size is based on the current font of the main window.

continued

continued

Field	Description
fc/bc	Foreground and background colors for this button. See Options for the available colors. If you specify only one color (omitting the slant), that color is used for both foreground and background. This field is not applicable for the option (P) button.
o	Type of button being created. See Options.
text	Text to appear in the button. Enclose text containing embedded spaces in apostrophes ('). If you do not specify text, place two apostrophes in this field.
vbh	Variable to capture the button handle. This must be a six-character I type variable.

Options

The following colors are available:

bla	black
whi	white
red	red
dred	dark red
blu	blue
dblu	dark blue
gre	green
dgre	dark green
cya	cyan
dcya	dark cyan
gra	gray
bro	brown
pin	pink
dpin	dark pink
yel	yellow
mag	magenta

Use the options listed below to designate the type of button to create. The **State** field contains the following values:

(0) Default. Displays the button or check box as unselected or unchecked.

(1) Displays the button or box as selected or checked.

(2) Displays the button in a third state. This state is available only with the 3 option.

Option	State	Button Type
P		Option (push) button.
C	(0) or (1)	Check box.
R	(0) or (1)	Command (radio) button.
A		Automatic button. Use in conjunction with C (AC), R (AR) and 3 (A3). Auto implies the state of the check box or command button; it automatically changes state when the user selects it prior to application notification. An auto command button also clears the state of all other command buttons in the same window.
3	(0),(1), or (2)	Three-state button
G		Grayed (disabled). This is the only option available when you specify a button handle in the *vwh* field.
B		Border (not available with P buttons).
F		Circle (for command button) or square (for check box) follows the text. This option applies to C,R, and 3 options only.
S		Send immediately. If you do not use this option, the system does not send the button until the current block is transferred; this results in increased performance. If you do not want the workstation to wait for the block transfer, however, you must use the S option.

Comments

- To request that the run wait for a selection from a button, and to determine the action based on that selection, use an INP run statement.
- The button remains on the screen until a CLS run statement closes it, or a window containing the button is closed, or the run terminates.
- If the user selects an automatic button, the software toggles the button. The software does not automatically toggle non-automatic buttons; use the BTN run statement to toggle them.
- You can define multiple automatic command buttons in a window, but only one can be in the selected state. Selecting any of the automatic command buttons automatically turns off all other automatic command buttons.

Example

This example uses the BTN statement to create a button, and the INP statement to continue at label 10 if the user presses the button.

```
@btn,,10,12,4,20,ar(1) 'Printer enabled' <pbutton>i6 .
@inp <pbutton>,(10) .
```

where:

`,,10,12`	Creates a new button at row ten, column twelve.
`,4,20`	Makes the button four lines tall and twenty characters wide.
`ar(1)`	Makes the button an automatic command button, already set.
`'Printer enabled'`	Displays the message 'Printer enabled' inside the button. Because this is an automatic command button, when the user presses the button it is toggled to the off state.
`<pbutton>i6`	Loads <pbutton> with the button handle.

CLS (Close Window)

Use the Close Window (CLS) run statement to close a MAPPER window. The statement closes the specified window, and any windows within that window.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@CLS[,vwh,vwh...] .
```

The following table describes the subfield:

Field	Description
vwh	Variable that contains the handle of the window to close. When you close a window handle, the run sets its corresponding variable to 0. Default = all windows.
	1100: Maximum = 40 entries.
	When you specify a window handle variable in a CLS statement, the statement closes that window and sets its corresponding window handle variable to 0. Window handles closed indirectly (their variables are not specified in the CLS statement but the windows are closed) are not set to 0.

Example

This statement closes the active windows that have window handles in <tools> and <shapes>.

```
@cls,<tools>,<shapes> .
```

DFC (Set Device Color)

Use the Set Default Color (DFC) run statement to set the default foreground and background colors when working with Designer Workbench run statements. The statement sets the default colors for the foreground and background. This statement does not set colors for the main window.

Note: *See Appendix J for information about the Designer Workbench environment.*

This run statement operates only with the following statements:

BTN (Button)	PIC (Picture)
EDT (Edit Box)	TXT (Text Box)
LST (List Box)	WIN (Window)

Format

@DFC, *fc/bc* .

The following table describes the subfield:

Field	Description
fc/bc	Default foreground and background colors for windows created with the following statements during this MAPPER session. See Options for the available colors. If you specify only one color (omitting the slant), that color is used for both foreground and background. BTN (Button) EDT (Edit Box) LST (List Box) PIC (Picture) TXT (Text Box) WIN (Window)

Options

The following colors are available:

bla	black
whi	white
red	red
dred	dark red
blu	blue
dblu	dark blue
gre	green
dgre	dark green
cya	cyan
dcya	dark cyan
gra	gray
bro	brown
pin	pink
dpin	dark pink
yel	yellow
mag	magenta

Example

This statement sets the default colors to white text on a blue background.

```
@dfc,whi/blu .
```

EDT (Define Edit Box)

Use the Define Edit Box (EDT) run statement to define an edit box and display it on the user screen. Use the Accept Input (INP) run statement to accept input from the edit box.

Note: See Appendix J for information about the Designer Workbench environment.

The edit box remains on the screen until a CLS run statement closes it.

Formats

```
@EDT,c,d,r[,l,q,vwh,row,col,rslz,cslz,fc/bc,o] [veh] .

@EDT[,"text",vwh,row,col,rslz,cslz,fc/bc,o] [veh] .
```

The following table describes the fields and subfields:

Field	Description
c,d,r	(Format 1 only) Report containing the text to be displayed.
l	(Format 1 only) Line in the report at which to start reading lines. Default = 1.
q	(Format 1 only) Number of lines to read from the report. Default = 1.
"text"	(Format 2 only) Text to appear in the edit box. If you do not specify text, place two quotation marks in this field. The system translates variables as follows: - If enclosed in apostrophes, no translation takes place. - If enclosed in quotation marks, variable is translated to its contents. *Note: The reverse slant (\ ) in literal text may be interpreted as a continuation character. Use the RSLANT$ reserved word (without apostrophes) to avoid misinterpretation. Or, place an at sign (@) in column one of the subsequent line.*

continued

continued

Field	Description
vwh	Variable containing an existing window or edit box handle. Default = main window. If you specify a window handle, the run places the edit box in that window, and all row and column coordinates are relative to that window. If you specify an edit box handle, the system assumes that you are changing that edit box in some way. If you do not specify a value in the ***row***, ***col***, ***rsiz***, ***csiz***, ***fc/bc***, ***typ***, or ***text*** fields, the system uses the original values.
row	Row number of the upper left corner of the edit box. Default = 1.
col	Column number of the upper left corner of the edit box. Default = 1.
rsiz	Vertical size of the edit box, in rows. Default = 0 (the edit box extends to the bottom of the window in which you place it and 'expands' and 'contracts' with that window). Specify a number to identify a permanent size for the edit box, based on the font size of the currently active font.
csiz	Horizontal size of the edit box, in rows. Default = 0 (the edit box extends to the right side of the window in which you place it and 'expands' and 'contracts' with that window). Specify a number to identify a permanent size for the edit box, based on the font size of the currently active font.
fc/bc	Foreground and background colors for this edit box. See Options for the available colors. If you specify only one color (omitting the slant), that color is used for both foreground and background.
o	Type of edit box being created. See Options. Once you have created an edit box of a certain type, you can change that type only by closing the box using the CLS statement and recreating it using the EDT statement. Default = left justified, fixed text length (the user cannot type past the borders of the box.)
veh	Variable to capture the edit box handle. This must be a six-character I type variable.

Options

The following colors are available:

bla	black
whi	white
red	red
dred	dark red
blu	blue
dblu	dark blue
gre	green
dgre	dark green
cya	cyan
dcya	dark cyan
gra	gray
bro	brown
pin	pink
dpin	dark pink
yel	yellow
mag	magenta

Use the options listed below to designate the type of edit box to create.

H — Scrolls horizontally. If a user tries to type past the right border of the edit box, the box scrolls to the right, allowing more room for text. The arrow keys allow horizontal movement within the box. Do not use with the Z option.

Z — Same as H, except that the run places a scroll bar beneath the edit box, for horizontal movement within the box. Do not use with the H option.

V — Scrolls vertically. If a user tries to type past the bottom border of the edit box, the box scrolls down, allowing more room in which to enter text. The arrow keys allow vertical movement within the box. Do not use with the T option.

T — Same as V, except that the run places a scroll bar to the right of the edit box, for vertical movement within the box. Do not use with the V option.

L	Left-justifies input text. Do not use with the C or R option.
C	Centers input text. Use in conjunction with the M option. Do not use with the L or R option.
R	Right-justifies input text. Use in conjunction with the M option. Do not use with the C or L option.
M	Accepts multiple lines of input. The run places input from this type of edit box into a result with the same cabinet and drawer as the current output area.
U	Displays input text in uppercase characters. Do not use with the O or P option.
O	Displays input text in lowercase characters. Do not use with the U or P option.
P	Makes input text invisible on the screen. Do not use with the U or O option.
D	Produces a drop box (an edit box with an arrow on the far right). The user clicks on the arrow to display a list of values beneath the box. The user clicks on a selection to place that selection in the edit box. When you use the D option, the H option is assumed. Do not use the D option with the M option. Supply the data to appear in the list of values in the *"text"* subfield (format 2) or the *c,d,r* subfields (format 1). Separate multiple items supplied in the *"text"* subfield with a slant (/). A number (enclosed in parentheses) following the D option identifies which item (of those listed between the quotation marks) appears as the default value in the edit box. If none is specified, no value appears.
B	Places a border around the edit box. If used, the row and column sizes of the edit box are one-half character larger than the sizes specified in the *rslz* and *cslz* subfields.
A	Accepts alpha data only.

N Accepts numeric data only.

S Specifies that the edit box is to be sent immediately. If you do not use this option, the system does not send the button until the current block is transferred; this results in increased performance. If you do not want the workstation to wait for the block transfer, however, you must use the S option.

Example 1

This example produces an edit box that allows input (a six-character password) from the user. After the user enters a password, the run continues processing at label 100. Because the variable <key> contains the window handle for the edit box, the insertion point is in the edit box while the run waits for input.

```
@txt,"Enter password",<gate>,2,2,1,15 <pass>i6 .
@edt,"",<gate>,2,18,1,6,,p <key>i6 .
@inp,<key> <key>,(100) .
```

Example 2

This statement produces a drop box. The values found within the quotation marks are those that appear in the list of values for the edit box, if requested. This list contains five values (1, 2, 3, 4, and 5). The d(3) option means that the edit box should appear with a default value of 3.

```
@edt,"1/2/3/4/5",v10,10,10,1,5,,d(3) v3i6 .
```

FON (Font)

Use the Font (FON) run statement to change the character font being displayed.

Note: See Appendix J for information about the Designer Workbench environment.

Format

@FON, *typ* , *name* , *size* [, *o*] .

The following table describes the subfields:

Field	Description
typ	Identifies the type of font. To restore the main window font, leave this and all remaining fields empty. Blank — Specifies the font for the main window. T — Temporary. Applies to any window other than the main session window, and remains active until the run terminates or until replaced by another font. R — Default font for the duration of the run. When the run terminates, the system reactivates the main font active when the run started.
name	Name of the font (for example, Helv). Enclose font names containing embedded spaces in apostrophes ('). Available fonts are listed under File –> Display items (Fonts selection) on the main menu bar.
size	Font size. Use one of the following values: *n* Point size of characters in the specified font. S Small version of the specified font. M Medium version of the specified font. L Large version of the specified font. When using the S, M, or L sizes, a corresponding size is selected by the software. These sizes are portable between PCs. Specific point sizes are not always portable.
o	Type of emphasis character. See Options.

Options

B Displays characters in bold type. Do not use with the I option.

I Displays characters in italic type. Do not use with the B option.

U Underlines characters. Do not use with the S option.

S Strikes through characters. Do not use with the U option.

Comments

- To display a list of available fonts, choose the Display submenu from the Setup Menu. Then choose the Main Window Fonts selection.
- Only nonproportional fonts are allowed for the main session window.
- If you change the main window font, the software adjusts row, column (*col*), row size (*rsiz*), and column size (*csiz*) values accordingly.

Example 1

This statement changes the font to the largest Courier style font available. Characters are underlined.

This font change does not apply to the main session window. The change is terminated when the run terminates or when it encounters another FON statement.

```
@fon,t,"Courier",l,u .
```

Example 2

This statement restores the main window font.

```
@fon .
```

HID (Hide Window)

Use the Hide Window (HID) run statement to remove a window (and all windows within it) from the screen. This statement is similar to the Close Window (CLS) statement except that the window still exists.

To display a window, use the Show Window (SHW) statement.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@HID[,vwh,vwh,...] .
```

The following table describes the subfield:

Field	Description
vwh	Variable that contains the window handle of the window to hide. Default = all windows.
	1100: Maximum = 40 entries.

Example

This statement temporarily removes the active windows whose handles reside in <shapes> and <sizes>.

```
@hid,<shapes>,<sizes> .
```

INP (Accept Input)

Use the Accept Input (INP) run statement to temporarily suspend a run, wait for input identifying which window was last selected, and specify where the run should continue processing, depending on a selection made by the user.

For the first format shown below, the statement suspends the run until it receives input. The run then continues at the specified label. For the second format, the statement does not suspend the run; it continues on the next line.

Note: See Appendix J for information about the Designer Workbench environment.

Formats

```
@INP[,awh,fxmt?,iwh vwh,(lab),vwh,(lab)...] .

@INP[,wh1,wh2,wh3...] .
```

The following table describes the fields and subfields:

Field	Description
awh	(Format 1 only) Variable that contains a window handle. This window becomes the active window and identifies the window in which the run places the insertion point while waiting for input. If this is initially a button handle, that button automatically becomes the default.
fxmt?	Transmit from the window whose handle you specified in the ***awh*** field, picking up input from that window? Y or N. If Y, you cannot specify the ***vwh*** or ***lab*** fields. Default = N (no action is performed.)
iwh	(Format 1 only) Variable that contains a window handle. If specified, the statement accepts input only from that window.

continued

continued

Field	Description
vwh	(Format 1 only) Variable that contains the handle of the selected window.
	1100: Maximum = 40 entries.
	If you do not specify a variable, the run waits for input, then continues at the next line.
lab	(Format 1 only) The label to go to if the run user selected the preceding window handle.
	1100: Maximum = 40 entries.
wh1,wh2,wh3...	(Format 2 only) One or more window handles. The statement accepts input from all windows whose handles are listed on this command. For this format, only one window handle can be that of a multiple-item list or multiline edit box. Specify that window handle last.
	1100: Maximum = 40 handles.
	For this format, only one window handle can be that of a multiple-item list or edit box. Specify that window handle last.

Comments

- If receiving input from a list box, and the input contains tab characters, load the variable or variables with the INVR1$ reserved word before executing the INP statement.
- Input from a multiline edit box is limited to 256 characters per line.
- The total number of input characters cannot exceed 6500. An error message is displayed and the input is truncated if the maximum is exceeded.
- You can use one of the following reserved words in the *vwh* field:

  WND$ Contains the handle of the main session window.

  ACTWIN$ Contains the handle of the active window.
- The INP statement overrides the effect of the Command Handler (CHD) run statement. If you are using an INP statement to receive input from a window, the system ignores any registered command handler routine.

Example 1

This statement determines where to continue the run, depending on a choice made by the user.

```
@inp <quit>,(100),<print>,(105) .
```

where:

<quit>	Specifies that variable <quit> contains the window handle.
(100)	Goes to label 100 if the user selects <quit>.
<print>	Specifies that variable <print> contains the window handle.
(105)	Goes to label 105 if the user selects <print>.

If the user selects neither window, the run continues at the next line.

Example 2

This example loads the input from window handle v1 into v10, from window handle v2 into v11, and from window handle v3 into v12.

```
@inp,v1,v2,v3 .
@chg input$ v10h16,v11i6,v12i3 .
```

LST (Define List Box)

Use the Define List Box (LST) run statement to define a list box and display it on the user screen. This box contains a list of items from which the user can make selections.

Note: *See Appendix J for information about the Designer Workbench environment.*

Format

```
@LST[,c,d,r,l,tf?,oname,vwh,row,col,rslz,cslz,fc/bc,
o,hl] caption [vlh] .
```

The following table describes the fields and subfields:

Field	Description
c,d,r	Report containing the list of items to be displayed in the window.
l	Line within the specified report at which to start transferring list items. Default = 3.
tf?	Transfer the report to the Designer Workbench workstation before processing the LST statement? Y or N. Default = N.
oname	Name of the object that contains the list items. If you do not specify a name here, it must exist on line 2 of the report specified in the c,d,r fields. See Appendix J for more information about naming objects.

continued

continued

Field	Description
vwh	Variable that contains an existing window or list box handle. Default = main window. If you specify a window handle, the list box is placed in that window. The coordinates specified in the ***row***, ***col***, ***rsiz***, and ***csiz*** fields are relative to that window. If you specify a list box handle, the system assumes that you are changing that list box in some way. If you do not specify a value in the ***row***, ***col***, ***rsiz***, ***csiz***, ***fc/bc***, or ***title*** fields, the system uses the original values.
row	Row number of the upper left corner of the list box. Default = 1.
col	Column number of the upper left corner of the list box. Default = 1.
rsiz	Vertical size of the list box, in rows. If 0, the list box extends to the bottom of the window in which it is placed, and expands and contracts with the window. Other sizes are permanent. Size is based on the font size of the currently active font. Default = 0.
csiz	Horizontal size of the list box, in columns. If 0, the list box extends to the far right of the window in which it is placed, and expands and contracts with the window. Other sizes are permanent. Size is based on the font size of the currently active font. Default = 0.
fc/bc	Foreground and background colors for this list box. See Options for available colors. If you specify only one color (omitting the slant), the statement uses that color for both foreground and background.
o	Type of list box to display. See Options.
hl	Number of the item to highlight in the list box. Default = no highlighting (except for the S option, which has a default of 1.)
caption	The caption of the list box. Enclose captions containing embedded spaces in apostrophes ('). If you do not specify a caption, place two apostrophes in this field. List boxes without captions cannot be moved on the screen.
vlh	Variable to capture the list box handle. This must be a six-character I type variable.

Options

The following colors are available:

bla	black
whi	white
red	red
dred	dark red
blu	blue
dblu	dark blue
gre	green
dgre	dark green
cya	cyan
dcya	dark cyan
gra	gray
bro	brown
pin	pink
dpin	dark pink
yel	yellow
mag	magenta

Use the options listed below to designate the type of list box to display.

S	Allows the user to select one item. The statement passes back to the run the value in the field following the selected item.
M	Allows the user to select more than one item. As each item is selected, it is highlighted. When all items have been selected, the statement passes those values back to the run as a result in the same cabinet and drawer as the current output area.
N	Similar to option M, but passes the item line numbers back to the run as a result in the same cabinet and drawer as the current output area.
F	Creates a thick frame around the window, enabling the user to expand or contract the window.
X	Enables the user to enlarge the window to full size.
U	Updates an existing list box. This indicates that something is being changed in an existing list box (for example, color or size). In this case, you must use the same variable in the *vwh* and *vlh* fields.
L	Adds a 'Locate' button and an edit box to the top of the list box, enabling the user to locate a target within the list box. When found, the target is highlighted (unless used with the M option) and centered within the list box. Do not use this option if you specified a window or list box handle in the *vwh* subfield.
D	DOS file list box. All directories are denoted by [directory name]. Double click on item to go into the directory. Double click on [--] to back up a directory. The directory name is displayed in the caption if specified. Similar to the M option.

Example

This example uses the LST statement to display a list box, and the Accept Input (INP) statement to determine where to continue the run after the user makes a selection from the box.

```
@lst,0,a,155,6,,,<win1>,7,12,10,20,,s,4 \
'List of Names' <win2>i6 .
@inp <win2>,(100),<win3>,(105) .
  .
  .
@100:chg input$ <name>s20 .
```

where:

`0,a,155,6,,`	Lists data in report 155a.
`<win1>`	Specifies the handle of the window in which to place the list box.
`7,12,10,20`	Displays the list box at line 7, column 12. Makes the list box 10 lines long and 20 characters wide.
`,,s,4`	Highlights the fourth item in the list box.
`List of Names`	Gives the list box the caption 'List of Names.'
`<list2>i6`	Places the list box handle in variable <list2>i6.
	If the INP statement detects that this window was used, the run continues processing at label 100, where the input is placed into <name>.

MBX (Define Message Box)

Use the Define Message Box (MBX) run statement to define a message box and display it on the user screen. The user must respond to each message before continuing with the application.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@MBX[,btyp,ityp,caption] message code,(lab) code,(lab)... .
```

The following table describes the fields and subfields:

Field	Description
btyp	Type of button (or buttons) displayed in the message box. See Options for the available buttons. Default = OK.
ityp	Type of icon displayed in the message box. See Options for the available icons. Default = no icon.
caption	Caption of the message box, up to 50 characters. Enclose captions containing embedded spaces in apostrophes ('). If you do not specify a caption, place two apostrophes in this field. Default = run name.
message	The message to be displayed in the message box, up to 800 characters. Enclose messages containing embedded spaces in apostrophes ('). If you do not specify a message, place two apostrophes in this field. To begin a new line, use a slant (/). See Appendix J for information about slant characters.
	1100: On OS 1100 MAPPER Systems, you can display up to 640 characters (including caption).
code,(lab)	Label to go to when the user selects the specified button. The first code listed is the default button. See Options for the available codes.

Options

The following button types are available for the *btyp* field:

OK	OK
OC	OK and Cancel
RC	Retry and Cancel
YN	Yes and No
YNC	Yes, No, and Cancel
ARI	Abort, Retry, and Ignore

The following icon types are available for the *ityp* field:

A	Asterisk
E	Exclamation point
S	Stop sign
Q	Question mark

The following button codes are available for the *code,(lab)* field:

O	OK
C	Cancel
A	Abort
R	Retry
I	Ignore
Y	Yes
N	No

Example

This statement creates a message box.

```
@mbx,oc,e,Error 'Try again?' 0,(5),C,(10) .
```

where:

`oc`	Displays two buttons — OK and Cancel — within the messages box.
`e`	Displays an exclamation point icon within the message box.
`Error`	Gives the message box the title 'Error'.
`'Try again?'`	Displays the message 'Try again?' within the box.
`0,(5),C,(10)`	If the user selects OK, the run continues at label 5. If the user selects Cancel, the run continues at label 10. The user must respond before doing any further processing.

MNU (Define Menu Bar)

Use the Define Menu Bar (MNU) run statement to define a menu bar and display it on the user screen. This menu bar provides available selections.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@MNU,type[,vwh] keyword[,"label",o,action/val] .
```

The following table describes the fields and subfields:

Field	Description	
type	Specifies type of menu bar, P (permanent) or T (temporary). Leave this field blank when using the RESET keyword. Default = T.	
vwh	Specifies a window handle. When you specify this handle, the statement attaches the menu bar to that window rather than defaulting to the main window.	
keyword	Specifies attributes of the menu bar:	
	MENUBAR	Names each position on the menu bar, from left to right. You may specify multiple MENUBAR keywords per MNU run statement. All menu bar keywords must specify an action to perform, or immediately be followed by a SUBMENU keyword.
	RESET	Restores the previous permanent menu bar. When using this keyword, omit the ***type***, ***label***, ***o***, and ***action/val*** fields.
	SUBMENU	Specifies that there is a pop-up menu associated with this pull-down menu item, and that the lines defining the popup menu immediately follow this subfield. You may specify multiple SUBMENU keywords per MENUBAR keyword. Terminate SUBMENU keywords with an END command.

continued

continued

Field		Description
	END	Signals the end of a SUBMENU or MENUBAR keyword. If there is a SUBMENU keyword that has not been closed with an END keyword, the statement considers END to be the end of that SUBMENU. If there are no open SUBMENU keywords, END signals the end of the MENUBAR keyword.
"label"	Names each position on the menu bar. Use with the MENUBAR and SUBMENU keywords only.	
o	Options field (see Options).	
action/val	Action to perform or value to provide. If you do not specify an action or value, provide one or more submenu definitions immediately following the MENUBAR command. If this is a permanent menu bar and there are no pull-down selections associated with this label, this subfield must contain an action that is valid from the control line.	

Options

Use the following options only on pull-down or pop-up menu definition lines. They are not valid on MENUBAR or submenu keywords that specify an action. Default = Unchecked and Enabled.

C — Displays item as checked. Do not use with U or S option.

D — Displays item as disabled (grayed). Do not use with the E or S option.

S — Displays a separator bar. The label field must contain two quotation marks ("") only. Do not use in combination with any other option.

M — Marks the end of one column of pull-down selections. The next item begins a new column. Do not use with the S option.

Comments

- A permanent menu bar is active until the statement encounters another menu bar. It does not terminate when the run terminates.
- A temporary menu bar is active until:
  - You replace it with another menu bar,
  - The run terminates, or
  - You restore a previous menu bar (see ***keyword*** subfield).
- The MENUBAR, RESET, and SUBMENU commands that define the menu bar also control the contents of the menu bar and the action taken by each selection.
- A MENUBAR keyword is terminated by any at sign (@) command, or by an END keyword not paired with a SUBMENU keyword. Also, a MENUBAR (with its SUBMENU) closes when another MENUBAR keyword begins.
- Specify one menu bar label per MENUBAR keyword. You can define up to 500 labels within one MNU run statement, up to 100 per menu bar item. The maximum number of characters per label is 79.
- To tell the user which character to use (in conjunction with the ALT key) when selecting an item via the keyboard rather than the mouse, place an ampersand (&) immediately preceding the character. When the user displays the menu bar, the character (called the hot key) is underlined. If multiple menu bar entries exist with the same hot key, the statement uses only the first entry.
- For permanent menu bars, if there are no pull-down selections associated with the menu bar name, you must provide an action in the action/val field.

- For temporary menu bars, use the INP statement to receive the contents of the action/val field. You can then reference the contents in reserved words INPUT$, INVAR$, and so on.
- If the run suspends via an OUT, DSP, OUM, or SC run statement, and the user selects a menu bar item, the statement considers the contents of the *action/val* field to be input from the home position of the screen and processes them accordingly.
- If you add a menu bar to a window, the menu bar takes space within that window. The actual height of the menu bar depends upon the type of monitor (VGA or EGA).

Example

This example uses the MNU statement to create a temporary menu with two entries separated by a separator bar. Selecting the 'Colors' entry produces a submenu that lists available colors. The colors are presented in two columns, the second starting at the entry 'Pink'. The entry 'White' is disabled and checked.

```
@mnu,t MENUBAR,"Options"
                "Border",,
                "",s,0
       SUBMENU,"Colors"
                "Blue",,1
                "Red",,2
                "Yellow",,3
                "Pink",m,4
                "Orange",,5
                "White",cd,6
```

PC (Run PC Program)

Use the Run PC Program (PC) run statement to execute a local PC application.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@PC,o command .
```

The following table describes the fields and subfields:

Field	Description
o	Action option. See Options.
command	Name of the application to be executed. Enclose the name in quotation marks if there are embedded spaces.

Options

B Executes the specified program and minimizes it as an icon.

W Waits for the main window of the specified program to close before continuing.

N Identifies the specified program as not being a Windows program (such as a DOS command).

Example

This statement executes the PC program called "clock".

```
@pc "clock" .
```

PCR (Transfer from PC)

Use the Transfer from PC (PCR) run statement to retrieve data from a Designer Workbench workstation into a MAPPER report.

Note: *See Appendix J for information about the Designer Workbench environment.*

Format

```
@PCR[,c,d,r,b?,tltln?,lab] oname .
```

The following table describes the fields and subfields:

Field	Description
c,d,r	Report to copy into.
b?	Does the file contain binary data? Y or N. Default = N. The run statement uses this field only when retrieving a non-repository data file.
tltln?	Use the specified object name as the title? Y or N. If you specify Y, the system writes the object name into line 2 of the report. Binary reports always insert a title or a blank line. Default = N (no title).
lab	Label to go to if the report does not exist.
oname	Object name from which to copy the data. If you do not specify an object name, place two apostrophes in this field. In this case, the name of the object must exist on line 2 of the report. See Appendix J for more information about naming objects.

Example

This statement copies a repository object named rose into report 63c32. The full object name 'rose.bmp()' is written to line 2 of the report and becomes its title.

```
@pcr,32,c,63,,y 'rose.bmp()' .
```

CW (MAPPER to PC)

Use the MAPPER to PC (PCW) run statement to transfer data from a MAPPER report to an object on the local PC.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@PCW,c,d,r[,l,lab] oname .
```

The following table describes the fields and subfields:

Field	Description
c,d,r	Report containing data to write.
l	Line number in the report at which to start copying data. Default = 3.
lab	Label to go to if the report does not exist.
oname	Object name to write to. If the object exists, it is overwritten. If it does not exist, it is created. If you do not specify an object name, place two apostrophes in this field. In this case, the name of the object must exist on line 2 of the report. See Appendix J for more information about naming objects.

Example 1

This statement copies data in report 153a0 to a file named 'a:\rose.bmp'.

```
@pcw,0,a,153 'a:\rose.bmp' .
```

Example 2

This statement copies data in report 153a0, starting at line 25, to the file. If report 153a0 does not exist, the run continues at label 100.

```
@pcw,0,a,153,25,100 '' .
```

PIC (Display Picture)

Use the Display Picture (PIC) run statement to display a picture or a graph within a window.

When displaying pictures, use a background color similar to the foreground (picture color) for aesthetic appeal.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@PIC[,c,d,r,tf?,oname,vwh,row,col,rsiz,csiz,fc/bc,
o] caption [vph] .
```

The following table describes the fields and subfields:

Field	Description
c,d,r	Report that contains the picture to be displayed.
tf?	Transfer the report to the workstation before processing the PIC statement? Y or N. Default = N.
oname	Name of the object that contains the picture. Each extension below identifies the type of device driver needed to process the picture. See Appendix J for more information about naming objects. .BMP bitmap .PCX PCX image .MGL MAPPER graphics primitives

continued

continued

Field	Description
vwh	Variable that contains an existing window or picture handle. Default = main window. If you specify a window handle, the statement places the picture in that window. All row and column coordinates are relative to that window. If you specify a picture handle, the system assumes that you are changing that picture in some way. If you do not specify a value in the *row*, *col*, *rsiz*, *csiz*, *fc/bc*, *o*, or *caption* fields, the system uses the original values. Once defined, you can change only the *row*, *col*, *rsiz*, *csiz*, *fc/bc*, *o*, and *caption* fields.
row	Row number of the upper left corner of the picture. Default = 1.
col	Column number of the upper left corner of the picture. Default = 1.
rsiz	Vertical size of the picture, in rows. Default = 0. This size is based on the font of the main window. If 0, the picture extends to the bottom of the window in which the user places it, and it expands and contracts with that window. If -1, the statement displays the picture using the actual dimensions as stored; place a -1 in the *csiz* field also.
csiz	Horizontal size of the picture, in columns. Default = 0. This size is based on the font of the main window. If 0, the picture extends to the right edge of the window in which it is placed, and it expands and contracts with that window. If -1, the statement displays the picture using the actual dimensions as stored; place a -1 in the *rsiz* field also.
fc/bc	Foreground and background colors for this picture. Only the background color is applicable when displaying pictures. See Options for the available colors.
o	Type of window to display. See Options.
caption	Caption for the picture. Enclose captions containing embedded spaces in apostrophes ('). If you do not specify a caption, place two apostrophes in this field. The system translates variables to their corresponding values. Pictures without captions cannot be moved.
vph	Variable that contains the picture handle. This must be a six-character I type variable.

Options

The following colors are available:

bla	black
whi	white
red	red
dred	dark red
blu	blue
dblu	dark blue
gre	green
dgre	dark green
cya	cyan
dcya	dark cyan
gra	gray
bro	brown
pin	pink
dpin	dark pink
yel	yellow
mag	magenta

Use the options listed below to designate the type of window to display.

F	Creates a thick frame around the window, enabling the user to expand or contract the window. Do not use with the B option.
B	Displays a border around the window. Do not use with the F option.
X	Enables the user to enlarge the window to full screen size.
I	Input picture. If the user clicks on this picture, the run resumes accordingly.

S — Step option. This option controls whether the run sends the picture data immediately or transfers it in blocks. If you specify a report in the *c,d,r* subfields, the run always sends the data immediately. If you do not specify a report, data is sent in blocks unless you use the S option.

P — Places a horizontal and vertical scroll bar on the picture. These scroll bars allow the user to pan vertically and horizontally through the picture.

Example

This statement displays a picture.

```
@pic,0,a,3,y,'rose.bmp()' 'Rose' <petals>i6 .
```

where:

`0,a,3`	Takes the picture from report 3A0.
`y`	Downloads the object before processing the PIC statement.
`'rose.bmp()'`	Specifies the object name of the picture.
`'Rose'`	Displays the picture with a caption of 'Rose'.
`<petals>`	Loads <petals> with the picture handle.

SHW (Show Window)

Use the Show Window (SHW) run statement to temporarily redisplay an existing window.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@SHW[,vwh,vwh,...] .
```

The following table describes the subfield:

Field	Description
vwh	Variable that contains the handle of the window to display. If the user displays a window containing other windows, the statement displays all windows. Default = all windows.

Example

This statement displays the windows whose handles reside in variables <toolist> and <gettool>.

```
@shw,<toolist>,<gettool> .
```

TXT (Define Text Box)

Use the Define Text Box (TXT) run statement to define a text box and display it on the user screen. Text boxes are for display purposes only; the statement returns no data.

Note: See Appendix J for information about the Designer Workbench environment.

Formats

@TXT[,c,d,r,l,q,vwh,row,col,rslz,cslz,fc/bc,o] [vth] .

@TXT[,"text",vwh,row,col,rslz,cslz,fc/bc,o] [vth] .

The following table describes the fields and subfields:

Field	Description
c,d,r	(Format 1 only) Report containing the text to be displayed.
l	(Format 1 only) Line in the report at which to start reading lines. Default = 1.
q	(Format 1 only) Number of lines to read from the report. Default = 1.
"text"	(Format 2 only) Text to appear in the text box. If you do not specify text, place two quotation marks in this field. The system translates variables as follows: - If enclosed in apostrophes, no translation takes place. - If enclosed in quotation marks, the statement translates the variable to its contents.

continued

continued

Field	Description
vwh	Variable containing an existing window or text box handle. Default = main window. If you specify a window handle, the statement places the text box in that window, and all row and column coordinates are relative to that window. If you specify a text box handle, the system assumes that you are changing that text box in some way. If you do not specify a value in the ***row***, ***col***, ***rsiz***, ***csiz***, ***fc/bc***, or ***text*** fields, the system uses the original values. Once defined, you can change only the ***c***, ***d***, ***r***, ***l***, and ***q*** fields of Format 1 (or the ***text***, ***row***, ***col***, ***rsiz***, ***csiz***, and ***fc/bc*** fields of Format 2).
row	Row number of the upper left corner of the text box. Default = 1.
col	Column number of the upper left corner of the text box. Default = 1.
rsiz	Vertical size of the text box, in rows. Default = 1. If a temporary font is active, this size is based on that font.
csiz	Horizontal size of the text box, in columns. Default = 7. If a temporary font is active, this size is based on that font.
fc/bc	Foreground and background colors for this text box. If you specify only one color (omitting the slant), that color is used for both foreground and background. See Options for available colors.
o	Options. See Options for the available options. Default = L.
vth	Variable to capture the text box handle. This must be a six-character I type variable.

Options

The following options are available:

bla	black
whi	white
red	red
dred	dark red
blu	blue
dblu	dark blue
gre	green
dgre	dark green
cya	cyan
dcya	dark cyan
gra	gray
bro	brown
pin	pink
dpin	dark pink
yel	yellow
mag	magenta

Use the options listed below to designate how to display the text.

L	Left-justified
C	Center
R	Right-justified
B	Border
S	Send immediately. If you do not use this option, the system does not send the button until the current block is transferred; this results in increased performance. If you do not want the workstation to wait for the block transfer, however, you must use the S option.
I	Input text. If the user clicks on this text box, the run resumes and is processed accordingly.

Example

This statement uses the TXT statement to display a text box.

```
@txt,"Printing in progress",<hal>,2,2,3,20,whi/red,c \
<print>i6 .
```

where:

`"Printing in progress"`	Displays the words "Printing in progress" in the box.
`<hal>`	Displays the text box in the window whose handle is contained in <hal>.
`2,2`	Creates the text box at row two, column two of the window.
`3,20`	Makes the text box three lines long and twenty characters wide.
`whi/red`	Colors the foreground of the box white, and the background red.
`c`	Centers the text in the box.
`<print>i6`	Places the window handle for this box in the variable <print>.

WIN (Define Window Display)

Use the Define Window Display (WIN) run statement to define a window and display it on the user screen.

Note: See Appendix J for information about the Designer Workbench environment.

Format

```
@WIN[,vwh,row,col,rsiz,csiz,fc/bc,o] caption [vwh] .
```

The following table describes the fields and subfields:

Field	Description
vwh	Variable to capture an existing window handle. If you use this field, all row and column coordinates are relative to the specified window. Default = main window. If you specify a window handle, you place a window within a window unless you specify the U option. The U option implies that you are changing an existing window in some way. If you do not specify a value in the ***row***, ***col***, ***rsiz***, ***csiz***, ***fc/bc***, or ***caption*** fields, the system uses the original values. Note: ***Although placing a window within a window is legal, it is of limited use, because without a mouse you cannot access controls directly. Navigation into and out of windows with a keyboard is not possible.*** Once you define a window, you can only change the ***row***, ***col***, ***rsiz***, ***csiz***, ***fc/bc***, and ***caption*** fields. If this subfield contains WND$ (the main window handle), only the caption field is used. That caption becomes the caption of the main window for the duration of the run. In this case, ***row***, ***col***, ***rsiz***, ***csiz***, ***fc/bc***, and ***o*** specifications are ignored.
row	Row number of the upper left corner of the window. Default = 1.
col	Column number of the upper left corner of the window. Default = 1.
rsiz	Vertical size of the window, in rows. Default = 5. This size is based on the current font of the main window.
csiz	Horizontal size of the window, in columns. Default = 10. This size is based on the current font of the main window.

continued

continued

Field	Description
fc/bc	Foreground and background colors for this text box. If you specify only one color (omitting the slant), that color is used for both foreground and background. See Options for the available colors.
o	Type of window to display. See Options for available options.
caption	Caption of the window, up to 50 characters. Enclose captions containing embedded spaces in apostrophes ('). If you do not specify a caption, place apostrophes in this field. Windows without captions cannot be moved. A caption bar is automatically added when an I, X, or F option is specified.
vwh	Variable to capture the new window handle. This must be a six-character I type variable.

Options

The following colors are available:

bla	black
whi	white
red	red
dred	dark red
blu	blue
dblu	dark blue
gre	green
dgre	dark green
cya	cyan
dcya	dark cyan
gra	gray
bro	brown
pin	pink
dpin	dark pink
yel	yellow
mag	magenta

Use the options listed below to designate the type of window to display.

C	Creates a child window. This window cannot move or exist outside the main window. Do not use with the O option.
O	Creates an overlapped window. This window exists wholly outside of the main window. Do not use with the C option.
F	Creates a thick frame around the window, enabling the user to expand or contract the window. Do not use with the B option.
B	Displays a border around the window. Do not use with the F option.
I	Enables the user to reduce the window to icon size.
X	Enables the user to enlarge the window to full size.
H	Hides the window from view until a SHW statement is requested.
S	Send immediately. If you do not use this option, the system does not send the button until the current block is transferred; this results in increased performance. If you do not want the workstation to wait for the block transfer, however, you must use the S option.
U	Updates an existing window. You can change only the *row*, *col*, *rslz*, *cslz*, *fc/bc*, and *caption* fields. In this case, use the same variable in both *vwh* fields.

Example

This statement uses the WIN statement to display a window.

```
@win,,10,40,12,20,,ixf 'BILLING' <winname>i6 .
```

where:

`,,10,40`	Displays a window at line 10, column 40 of the main session window.
`12,20`	Makes the window 12 lines long and 20 characters wide.
`ixf`	Frames the window so that the user can resize it. Also, give it maximize and minimize capabilities.
`'BILLING'`	Gives the window the caption 'BILLING'.
`<winname>i6`	Loads the variable <winname> with the window handle.

WSF (Display Workstation Form)

Use the Display Workstation Form (WSF) run statement to display and control a Designer Workbench form. Forms, which are created by using the Designer Workbench Forms Designer, consist of an enclosing window containing one or more controls (fields, list boxes, and buttons).

Note: See Appendix J for information about the Designer Workbench environment.

Condition

You must have the Forms Designer installed to create forms for the WSF run statement.

Format

```
@WSF,{c,d,r|"oname"}[,vwh,focus,o] val1,val2,...,valn [vfh] .
```

The following table describes the fields and subfields:

Field	Description
c,d,r	Report containing the form to be displayed.
oname	Name of the form to be displayed. See Appendix J for information about naming repository objects. The name must be enclosed in quotation marks (").
vwh	Variable that contains the handle of a currently open form. Use to send new value or status data to a form.
focus	Data order of the control that is to get the input focus. If you do not specify focus on a form open request, the statement gives focus to the control with the lowest data order. In all other cases, focus remains at its current location.
o	Form control options. See Options.
val1... *val2...*	Value or status data list to be passed to the controls. The first item goes to the control with data order 1, the second to the control with data order 2, and so on. **1100:** Maximum = 40 entries.
vfh	Variable to capture the form handle. Use when opening a form.

Options

C Specifies that a form caption is being supplied. The first data value is placed into the caption bar of the form.

H Opens the form but keeps it hidden (invisible). The form may be subsequently shown via the SHW run statement.

O Overlays the values or statuses of selected form objects.

S Modifies the status of form objects. The values given to the form determine how the objects are affected.

Comments

- The statement displays (opens) a form. The application may subsequently send new data values to the controls or modify their statuses (enable/disable, hide/show). User input from the form is captured using the reserved word INPUT$.
- Use the INP (Accept Input) run statement to enable the user to interact with the form.
- Each control (or control group in the case of buttons) interacting with the application must have a unique data order. This is an integer value that identifies which value in the WSF value list the control receives. It also identifies which INPUT$ variable receives the value of the control when the form sends input to the application. The statement assigns data orders to controls via the Designer Workbench Forms Designer.
- See Appendix J for information on naming objects.

Control Types and Their Data Values

Table 7-24 describes the control types that send and receive data, along with the meaning of their data values.

Table 7-24. Control Types

Control Type	Meaning of Data Values		
Fields	The text displayed in the field.		
List Boxes	Output	Name of the data object whose contents are to be displayed in the box.	
	Input	Contents of the data field or fields from the selected list box item.	
Buttons	In general, the selection state of the button group. For identification, the run statement numbers group members from the top or left (depending on whether you lay out the buttons vertically or horizontally), starting at 1. The exact meaning of the state data depends on the button style:		
	Command buttons	Output	The identification of the button that is to be the current default command button of the form.
		Input	The identification of the selected button.
	Option buttons	Output	The identification of the button that is to be initially selected.
		Input	Same as output, but indicating which button is currently selected.
	Check boxes	Output	A series of 0s and 1s that identify which box or boxes are to be initially selected. 1 = selected.
		Input	Same as output, but indicating which box or boxes are currently selected.

Status Values

When the run statement S option is specified, valid status values are as follows:

Status Value	Description
D	Disable the control
E	Enable the control (default)
H	Hide the control
S	Show the control (default)
Fields and list boxes	A single value from the above
Buttons	A series of status values from the above

WSF Statement Examples

Examples 1 and 2 use a form containing the following objects:

Object Type	Members	Data Order
Field		1
List Box		2
Field		3
Option Button	4	4
Check Box	6	5
Push Button	3	6

Example 1

This statement causes the subsequent actions to occur:

```
@wsf,20,b,10 'Acme','2012.dat(*)',,2,011000,1 v1i6 .
```

- The field with data order 1 displays the text "Acme".
- The list box displays contents of object "20I2.dat(*)".
- The field with data order 3 displays the default value supplied when the form was designed. If none was supplied, the field is blank.
- In the command button group, button 4 is selected.
- In the check box group, the second and third buttons are selected.
- In the command button group, the first button is designated as the form's default command button (probably the OK button).

Example 2

This statement causes the subsequent actions to occur:

```
@wsf,,,,v1,2,os ,h,,' dd',,eee .
```

- Modifies only the objects that receive a new status value. The status of all others remains unchanged.
- Hides the list box (data order 2).
- Disables the second and third buttons in the option button group.
- Enables all three command buttons (data order 6).

A

Creating a MAPPER Application

Objective

The objective of this appendix is to provide a general overview of creating a MAPPER application with the Designer Workbench. This appendix will also supply information on the documentation and online assistance for creating a MAPPER application.

MAPPER Software and Designer Workbench

Designer Workbench and MAPPER work together to download and upload information necessary to run MAPPER applications. A MAPPER application consists of:

- MAPPER run(s)
- MAPPER report(s)
- Forms and associated objects created with the DW Forms Designer

Forms can be created two ways:

- DW Forms Designer
- MAPPER DW run statements

Form objects have a data order to identify the order in which input is passed to the host. Objects also have an associated status to specify how an object is displayed to the end user.

MAPPER Software and Designer Workbench

- **Work together to download/upload information necessary to run MAPPER applications**

- **MAPPER applications consist of**
  - MAPPER run(s)
  - MAPPER report(s)
  - Forms

- **Forms created two ways**
  - DW Forms Designer
  - MAPPER run statements

- **Form objects have**
  - Data orders identifying order of input passed to host
  - Associated status specifying how objects are displayed

MAPPER Application Development

There are five basic steps in creating a MAPPER application using components of Designer Workbench:

1. Planning the application
2. Using the Forms Designer to create forms
3. Creating data files for the necessary MAPPER reports
4. Writing the MAPPER code for the application
5. Upload, test, and edit forms

MAPPER Application Development

- **Five steps in creating MAPPER applications using DW**
  - Plan the application
  - Use the Forms Designer to create forms
  - Create data files for necessary MAPPER reports
  - Write the MAPPER run
  - Upload, test, and edit forms

Theory of Application Processing

- **Developer creates forms and writes MAPPER code**
- **Developer uploads form - objects to MAPPER host**
- **User runs the application**
  - Form is accessed
  - Application downloads form to DW
  - Form is stored and called again from Repository
- **User enters data on to form and transmits**
  - Information is processed by the host
  - Output form is displayed

Information Source

When preparing to create a MAPPER Application using Designer Workbench, additional information can be obtained from the following source:

Designer Workbench Developer Training Guide (7831 9738-000)

- Section 1. Introducing Designer Workbench and the Forms Designer
  - Overview of Designer Workbench
  - Overview of the Forms Designer

- Section 2. How MAPPER and DW work together
  - Overview of MAPPER and DW
  - How MAPPER and DW interact

- Section 3. Using the Forms Designer
  - Starting the Forms Designer
  - Specifying Form Characteristics
  - Adding Objects on forms
  - Specifying Labels for form objects
  - Editing a form
  - Getting help

- Section 4. Creating the Sample Application
  - Exercise 1: Creating the Sample Input form
  - Exercise 2: Creating the Sample Output form

- Section 5. Creating the Sample Application Data Files and Runs
  - Exercise 3: Creating the Sample MAPPER Report Data
  - Exercise 4: Creating the Sample MAPPER runs

- Section 6. Testing the Sample MAPPER Application
  - Testing a Designer Workbench Form
  - Exercise 5: Uploading Data to the MAPPER Host
  - Exercise 6: Testing Your Application

- Appendix A. The Workstation Form (WSF) and PC Read (PCR) Run Statements
  - WSF Statement syntax
  - PCR Statement syntax

Information Source

- **Designer Workbench Developer Training Guide**

  - Section 1 Introducing DW and the Forms Designer
  - Section 2 How MAPPER and DW work together
  - Section 3 Using the Forms Designer
  - Section 4 Creating the Sample Application
  - Section 5 Creating the Sample Application Data Files and Runs
  - Section 6 Testing the Sample MAPPER Application
  - Appendix A The Workstation Form (WSF) and PC Read (PCR) Run Statements

Creating Forms with MAPPER Run Statements

Forms can be created by using MAPPER run statements. The run statements used to create form objects are listed below:

- @WIN Define Window Statement
- @SHW Show Window Statement
- @CLS Close Window Statement
- @HID Hide Window Statement
- @INP Accept Input Statement
- @BTN Define Button Statement
- @EDT Define Edit Box Statement
- @LST Define List Box Statement
- @MBX Define Message Box Statement
- @WSF Display Workstation Form Statement
- @PIC Display Picture Statement
- @TXT Define Text Box Statement
- @FON Font Statement
- @DFC Set Default Color Statement
- @MNU Define Menu Bar Statement
- @PCR Transfer data from PC file to MAPPER report
- @PCW Data Transfer from MAPPER to PC
- @PC ~~Phrase Change~~ RUN PC Program Statement

When deciding whether to use the Forms Designer or MAPPER run statements to create a form, consider the following:

- Forms Designer
  - Easy to use
  - Pictures have to exist in the Repository
  - Less interactive

- MAPPER Run Statements
  - More difficult to code
  - More versatile and interactive
  - Pictures can easily be retrieved from host

Use the WBX run as a tool for learning the MAPPER run statements. This run contains many interactive examples of all of the run statements for creating form objects.

Creating Forms with MAPPER Run Statements

- **Forms created by using**
  - MAPPER run statements
  - Forms Designer
- **Forms Designer vs MAPPER run statements**
  - Forms Designer
    - Easy to use
    - Pictures have to exist in the Repository
    - Less interactive
  - MAPPER Run Statements
    - More difficult to code
    - More versatile and interactive
    - Pictures can easily be retrieved from host
- **WBX run is a tool for learning the MAPPER run statements**

WBX Run Example

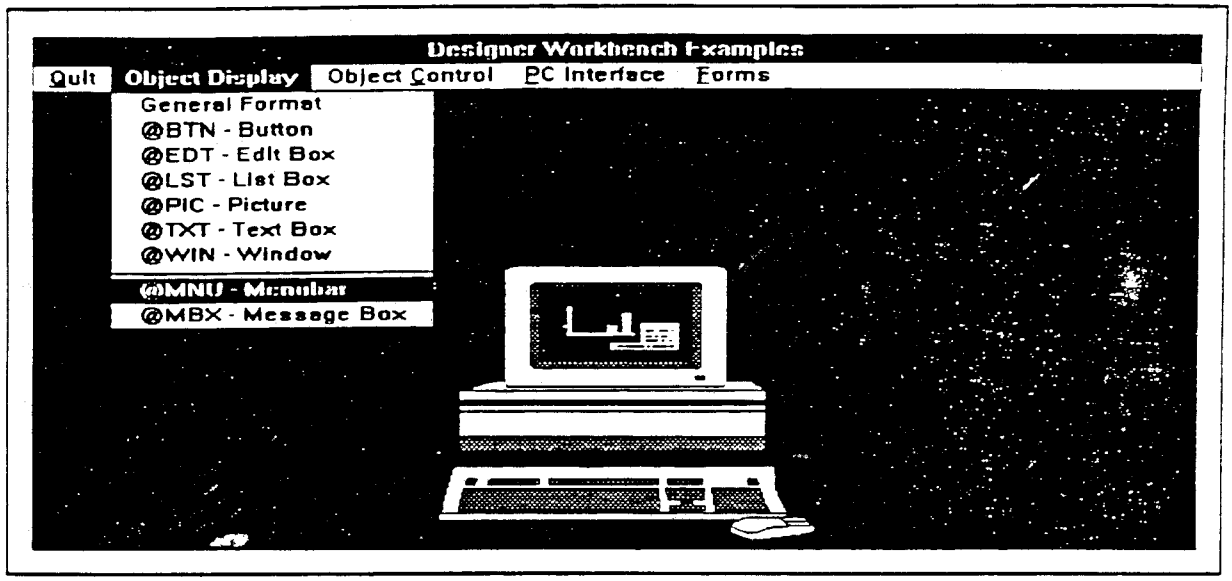
The following pages provide an example of the WBX run. This run is included in releases of MAPPER 4R2 and 36R1

Visual 1: Enter WBX on line 0 or select WBX from the Runs menu bar selection. The WBX run will display a customized menu bar. Make selections from the menu bar for specific examples. In this example, we select @MNU for an example of how the Menubar run statement works

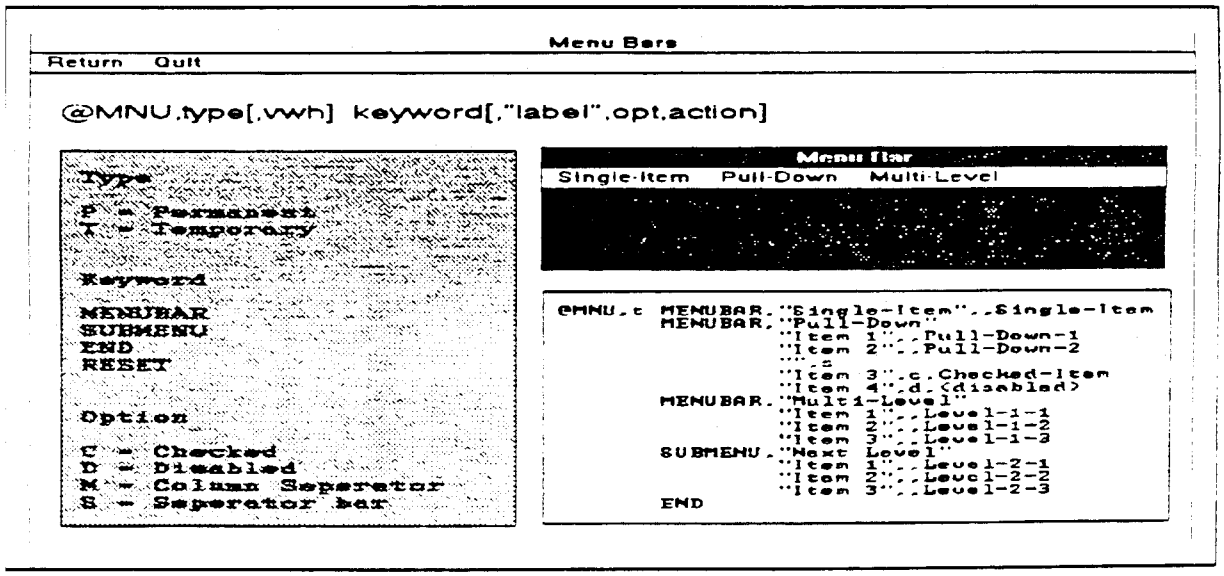
Visual 2: A screen for the MNU run statement is displayed It contains the syntax for the run statement as well as acceptable entries for sub-fields. It also has a built-in interactive example.

WBX Run Example

- Enter WBX on line 0, select 'Object Display', select '@MNU'



- @MNU example screen is displayed



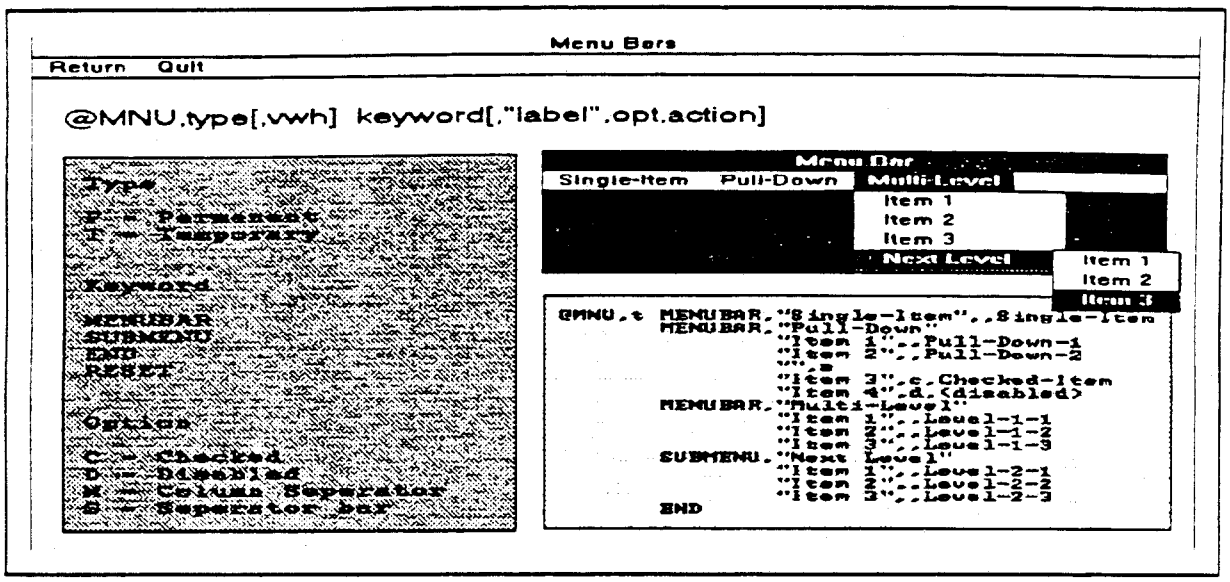
WBX Run Example

Visual 1: To try the built-in example, select one of the mini-menubar choices. In this example we will select 'Multi-Level' and 'Item 3'

Visual 2: The WBX run responds to the selection by highlighting the run code that would be necessary to handle the transaction.

WBX Run Example

- **Make selection from mini-Menubar**



- **Run code to handle transaction is highlighted**