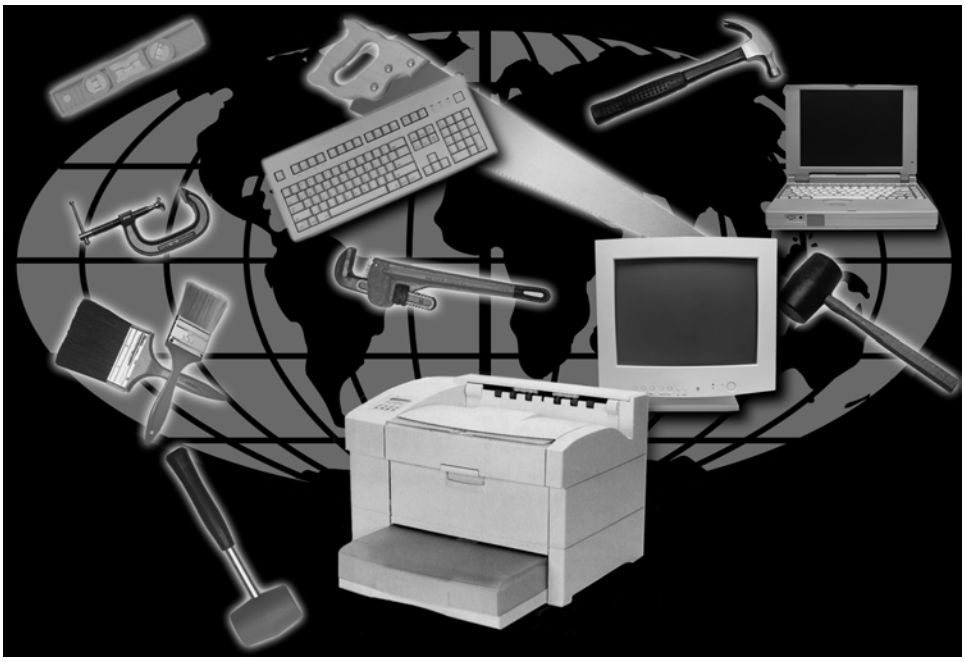


COBOL FormPrint
Windows Form Printing for COBOL
Version 4.1 User Guide

Flexus
P.O. Box 640
Bangor PA 18013-0640
U.S.A.

Voice: 610-588-9400
Fax: 610-588-9475
E-Mail: info@flexus.com
WWW: http://www.flexus.com
Anonymous ftp: ftp.flexus.com/incoming
Production ftp: ftp.flexus.epix.net

TABLE OF CONTENTS

PART A

Getting Started

CHAPTER A1
Before You Begin

What is COBOL FormPrint?

COBOL FormPrint is a tool used by COBOL programmers to develop and maintain contemporary print forms for their programs. It allows the complexities of such forms to be somewhat isolated from a COBOL program, yet still enables a COBOL programmer to have close control over the form directly from a program using standard COBOL code.

The COBOL FormPrint Form Editor

COBOL FormPrint provides a Form Editor for designing and developing your printer form.

The Runtime System

For implementing your print form, COBOL FormPrint provides a powerful runtime system (also used by the Form Editor itself) accessed through simple call statements. Sample call statements and a copy file containing interface properties can be generated using the Form Editor's own code generator.

A STEP-BY-STEP APPROACH TO USING COBOL FormPrint

Here is a step-by-step outline of a typical COBOL FormPrint Development session:
1. Define your form using the COBOL FormPrint Form Editor
2. Generate the data area for your form using the Code Generator included with the COBOL FormPrint Form Editor
3. Include the generated data area in the printing subroutine within your COBOL application
4. Modify the logic in your printing subroutine to pass data to the new data area
5. Call QPR (COBOL FormPrint) for printing each page

WHAT YOU NEED TO USE COBOL FormPrint

1. The COBOL FormPrint Form Editor.
2. QPR.DLL (The COBOL FormPrint Runtime System)

WHAT YOUR CUSTOMER NEEDS TO USE COBOL FormPrint

1. QPR.DLL (The COBOL FormPrint Runtime System)
2. All forms files containing forms used in your application.
3. All bitmap image files included in the forms used in your application.
4. If your forms include radio buttons, your customer will need the bitmap image files for selected and unselected radio buttons. These files are named RON.BMP (selected radio button bitmap image) and ROFF.BMP (unselected radio button bitmap image).
5. The FONTS.QPR file which contains information about form fonts and color ID's used in your FormPrint forms.

TERMINOLOGY

Here is a formal definition of some of the terms used in the rest of this guide. Some of these terms have already been mentioned above.

Form file

This is the file in which COBOL FormPrint stores forms. This file may be manipulated with regular operating system commands.

Form

This is the actual form definition that your program will send to the printer. Forms are defined using the Form Editor.

Field

An object defined within a form for the purpose of displaying data on a printed form. Fields are defined using the Form Editor. Static fields are used for forms text which does not generally change, labels for non-static fields and other printed text which generally remains constant. Non-static fields are used for printing variable data and printing non-static data.

Group

An object used to logically group non-static fields within a form. Groups are defined using the Form Editor.

Repeat Group

An object used to allow a non-static field or group of fields to be defined once and used multiple times within a form. Repeat groups are used to create vertical and horizontal arrays.

Color

The attribute assigned to a field, group or form setting the foreground and background colors of the object. Colors are defined using the Form Editor.

Font

The attribute assigned to a field or form controlling the printed output for that object. Fonts are defined using the Form Editor.

WHAT'S IN THE REST OF THIS GUIDE?

The COBOL FormPrint User Guide is divided into 5 parts. Each part has been assigned a letter and represents a major User Guide topic. The remaining 4 parts are:

Part B, The COBOL FormPrint Form Editor.
Part C, Programming with COBOL FormPrint.
Part D, FormPrint Programming Functions
Part E, FormPrint Properties

A Summary of PART B

PART B, The COBOL FormPrint Form Editor, provides a basic introduction to the Form Editor and how to use the editor in general. If you encounter difficulty in creating, deleting, moving or changing the size of fields in the Form Editor, you should refer to Chapter B2. This chapter explains the standard procedures to use when working with fields in the form you are creating. Part B will help significantly reduce your learning curve when using the Form Editor.

A Summary of PART C

PART C, Programming with COBOL FormPrint, provides an overview of the concepts involved when using COBOL FormPrint. Part C is helpful when you first start using COBOL FormPrint.

This section of the manual explains in general terms the standard procedures you will use when using COBOL FormPrint. Topics covered include how to select the printer, pass data to forms, and eventually send forms to printer, close and end your printer session.

A Summary of PART D

PART D, FormPrint Programming Functions, lists the various FormPrint functions used to control the printing from your COBOL program.

A Summary of PART E

PART E, FormPrint Properties, lists the various FormPrint data variables passed to FormPrint by your COBOL program. The entries in the various chapters of PART E will list the Properties as well as the functions needed to set the item in your program.

Appendix A

Appendix A lists the Environment Variables used to configure COBOL FormPrint.

Appendix B

Appendix B provides techniques for organizing forms within your forms files. It also describes the QPRCHECK utility which is used to compress and update forms files.

Appendix C

Appendix C lists the various error codes returned by the programming functions used with COBOL FormPrint.

Appendix D

Appendix D describes the COBOL FormPrint Code Generator template file, called UIB.CBX. This template file can be modified to produce code that may be more suited to your own needs.

Appendix E

Appendix E lists the property ids used with the GET-PROPERTY and SET-PROPERTY functions and the property keys used with the Code Generator.

INSTALLATION OF COBOL FormPrint and THE FORM EDITOR

COBOL FormPrint and the Form Editor are shipped as an executable (.EXE) file. This allows you to start the installation using the Windows Program Manager or the Windows 95/98 Start/Run feature to install the package. If you are updating, please back up the FONTS.QPR file prior to installation.

SYSTEM REQUIREMENTS

COBOL FormPrint requires approximately 2.4 megabytes of hard disk space.

GETTING HELP

The COBOL FormPrint Users Guide should provide answers to most of your questions. Please study the manual carefully before calling for assistance. If after reviewing the manual, you are still unsure of the solution to your problem, please call the telephone number on the first page of this book. You will also find the facsimile number, e-mail address and the location of our ftp site

Note: If your problem is complex or if we are unable to duplicate the problem, you may need to send us a small sample application which will demonstrate the problem. If so, please remember to send the following:

1. Only small sample programs (source code in addition to executable files as we must be able to compile the files in our environment. The source files and executables should not contain any file handling)

2. Your FONTS.QPR file.

3. The forms file(s) used by the sample program.

4. The version number of COBOL FormPrint.

CHAPTER A2
An Introduction to Windows Printing

When you use a printer in the Windows environment, you are actually initiating a complex interaction involving the printer device driver and the Windows Print Manager or Spooler. When a program wants to begin using a printer, Windows loads the device driver for that printer to process printer requests. Output to the printer will be held by the Print Spooler and directed to the appropriate physical output port. This means that your program will most often finish before all pages have been physically printed.

In most cases, Windows allows you to configure the printer driver to suit your needs. For example, you can specify that all printer output is to be redirected to a disk file rather than sent to the printer. You can do this either by using the Windows Control Panel so that a particular setting applies to all print programs, or by using the SELECT-PRINTER function so that the settings apply only to the current FormPrint program.

Occasionally, you may need to reconfigure the driver to permit the print output to be processed properly. For example, you may have to reset the printer timeout settings to allow a large print job to be handled. This must be done through the Windows Control Panel.

PART B

The COBOL FormPrint Form Editor

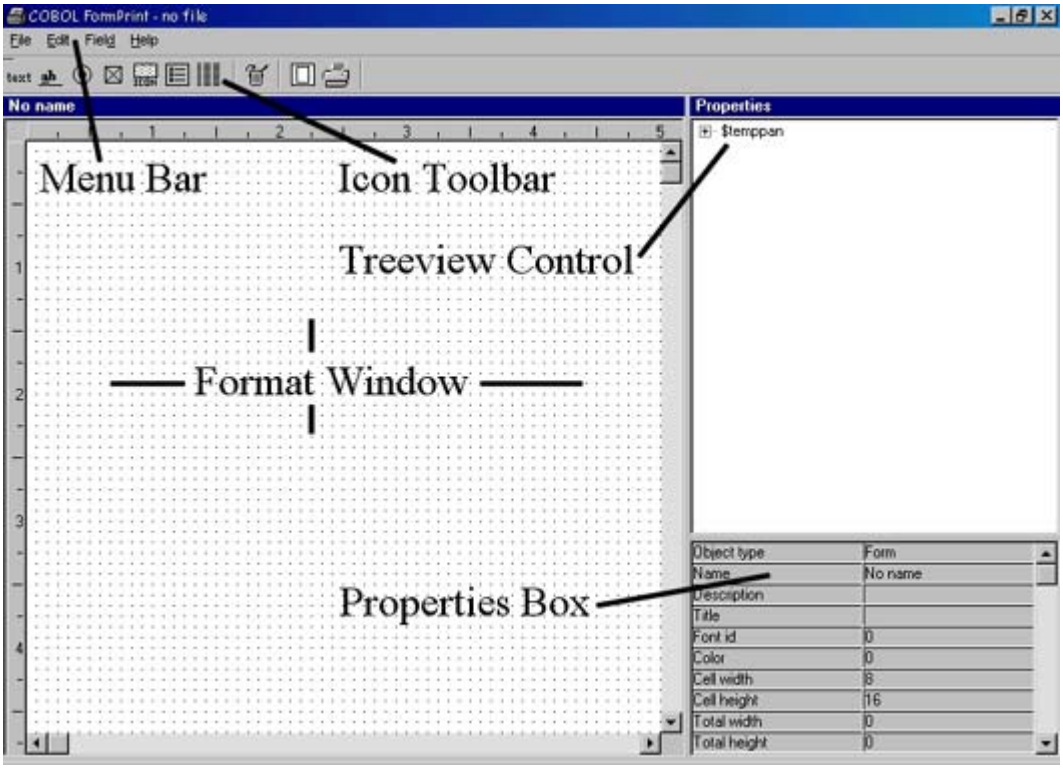
CHAPTER B1
An Introduction to the COBOL FormPrint Form Editor

Launch the COBOL FormPrint Form Editor

1. Invoke COBOL FormPrint from the Windows environment by clicking on the FormPrint icon.

2. The COBOL FormPrint logo window will display for a few moments. The FormPrint Form Editor Tool will then display.

Components of the Form Editor



As you can see in the previous image, there are five main components of the COBOL FormPrint Form Editor. The components are as follows:

- The Format Window
- The Icon Tool Bar
- The Menu Bar
- The Treeview Control
- The Properties Box

The Interrelationships between the Five Components

Each of the five components will be addressed individually later.

The Format Window is your work area. This is the area where you:
a) specify the location of your form fields,
b) resize the default form fields to suit your needs and

c) combine various form fields to complete the appearance of your form.

The Icon Tool Bar is used to select various form fields for placement in the Format Window. The Icon Tool Bar should significantly speed completion of the user interface design if you prefer to use a mouse.

For example, assume that you would like to place a data field on the proposed form. The first step to accomplish this task would be to move the mouse pointer to the second icon on the Tool Bar and click once. You would then move the pointer to the desired location in the Format Window and click again. A default data field will then appear in the Format Window automatically.

As mentioned previously, you could then easily change the location or size of the default field within the Format Window.

The Menu Bar is also available to create form fields in the Format Window. The Menu Bar also provides access to the same form design tools which are available on the Icon Tool Bar.

For creation of standard form fields, it is suggested that you use the Icon Tool Bar for faster form design, but if you prefer not to use a mouse when designing the form, the Menu Bar provides you with all of the same form design facilities.

The Menu Bar is also used for accessing form file management tasks and tools not available on the Icon Tool Bar such as opening form files, saving your work to disk and modifying the size of the form.

The Treeview Control can be used for several purposes. When a form file is opened, forms listed on the Treeview Control may be opened in the Format Window of the COBOL FormPrint Form Editor by clicking on the form name listed in the Treeview Control and "dragging" the form into the Format Window.

In addition, when updating the properties of fields in a form, it may be easier for you to place focus on a specific field by selecting it from the Treeview Control than selecting it from within the Format Window.

The Properties Box allows you to specify the properties of the form fields once they are placed in the Format Window. In the example of the data field just created, the developer could easily specify:

a) a COBOL field name,
b) initial value for the data field,
c) COBOL picture clause and
d) whether the field is an alphanumeric, numeric or date field.

This is all accomplished by typing the appropriate values within the appropriate property in the Properties Box.

The properties within the Properties Box will dynamically change depending upon the specific type of field on which you are setting the properties.

In order to enter or select a property for a specific field or field group, the cursor in the Format Window must be on the field or field group for which you want to establish the property.

Using the COBOL FormPrint Form Editor

The Form Editor may be navigated with the keyboard, the mouse or both the keyboard and the mouse. Although the keyboard must be used when specifying field captions, field formats and initial values, using the mouse as much as possible will greatly speed the process of form development.

Accessing the COBOL FormPrint Menu Bar

With the Mouse pointer

Click on the desired menu bar option. The corresponding pull down menu will appear, allowing you to click on the desired choice.

With the Keyboard

Use the (ALT) or (F10) key to activate the menu bar. Press the (DOWN ARROW) or (ENTER) key to display the corresponding pull down menu. Use the (ARROW) keys to move to the desired choice and select the choice using the (ENTER) key.

Accessing the COBOL FormPrint Icon Tool Bar

The Icon Tool Bar is used for several purposes in the COBOL FormPrint Form Editor. The Icon Tool Bar can be used to create fields and field groups in the Format Window as well as delete them. Accessing the Icon Tool Bar can be accomplished as follows:

With the Mouse pointer

Position the mouse pointer on the desired icon and click the left mouse button once. The pointer symbol may change, depending upon the specific icon selected. See the section on Pointer Symbols in the COBOL FormPrint Icon Tool Bar below for more information on those icons which change the pointer symbol.

With the Keyboard

The Icon Tool Bar must be activated with the mouse. All tools available on the Icon Tool Bar are also accessible through the Menu Bar, if you prefer to use the keyboard instead of the mouse.

Pointer Symbols in the COBOL FormPrint Icon Tool Bar

The Icon Tool Bar uses two different pointer symbols. The default pointer symbol, which is a diagonally pointing arrow, is used to initially select the icon.

After the Icon has been selected, the pointer will change to an action pointer for those icons which are used to:

a) create a field in the Format Window, or
b) select a field in the Format Window for further definition.
When the pointer symbol changes to the action pointer (vertically pointing arrow or a four-way pointing arrow, depending upon your operating environment), it is an indication that action is needed in the Format Window. The action to be taken depends on the specific icon selected.

The pointer will remain as a diagonally pointing arrow for those icons which require further definition of field or group properties in the Properties Box. These properties, or rules, are set in the appropriate property of the Properties Box.

Actions to be Taken After Selection of Specific Icons

1. The icons which change to an action pointer when selected and the specific action to be taken include:

Icon	Action to Be Taken
Static Text	Create Static Text in Format Window
Custom Field	Create Custom Field in Format Window
Radio Button	Create Radio Button in Format Window
Check Box	Create Check Box in Format Window
Bitmap	Create Bitmap in Format Window
Group Box	Create Field Group Box in Format Window
Repeat Field	Create Repeat Field Box in Format Window
Delete Field	Delete Field in Format Window
Page Setup	Select Paper Size and Orientation
Print Sample	Print a Sample Copy of the Form

Accessing the COBOL FormPrint Properties Box

With the Mouse pointer

Using the mouse, position the pointer on the field in the Format Window for which you want to establish a specific characteristic. Click the left button once to position the cursor. The Properties Box will then display the properties for that specific field.

If you are setting general form-level properties, position the pointer in an area of the form where no fields, groups or repeat groups exist and click the left button once to position the cursor. You may also double click or press the (ENTER) key on the panel description located in the Treeview Control. The Properties Box will then display form-level properties.

You may also double click or press the (ENTER) key on the form description in the Treeview Control.

With the Keyboard

When the cursor is in the Format Window, you may access the Properties Box by pressing the (ESC) key once. The Form Editor will attempt to maintain the cursor in the last property accessed when you toggle back and forth between the Format Window and the Properties Box. This works best when going from one field to the next field of a similar type. When you access a series of dissimilar fields, the cursor may jump to the top property in the Properties Box, because the properties in the Properties Box change depending upon the field which currently has focus in the Form Editor.

To access other properties in the Properties Box, press the (UP ARROW) or (DOWN ARROW) key to move the cursor.

Depending upon the specific property, some properties in the Properties Box are entry fields. Properties may be simply typed into these fields. Other properties are selection fields which provide a variety of selections for certain types of field behavior.

Pointer Symbols in the COBOL FormPrint Properties Box

The Properties Box uses two different pointer symbols. The default pointer symbol, which is the I-Beam pointer is used for data entry fields.

The second pointer symbol, the diagonally pointing arrow, is used to make selections in those properties within the Properties Box which provide two or more selectable options in a drop down list.

Accessing the COBOL FormPrint Format Window

With the Mouse Pointer

Using the mouse, position the pointer in the desired location within the Format Window. Click the left mouse button once to position the cursor. To move the cursor to an area of the Format Window which is not currently visible, click the left button on the horizontal or vertical scroll bar within the Format Window.

You can also scroll the Format Window by clicking and holding the left mouse button while dragging the pointer left, right, up or down. The Format Window should automatically scroll when the pointer is dragged outside of the Format Window.

With the Keyboard

If the cursor is already within the Format Window, you may locate the cursor by using the (ARROW) keys. If the cursor is in the Properties Box, press the (ESC) key to locate the cursor in the Format Window. Please note that pressing the (ESC) key while the cursor is in the Properties Box will cause you to lose any changes you may have made in the last field in which the cursor was residing when the (ESC) key was pressed.

If the Menu Bar has been activated and a menu bar option is highlighted, pressing the (ESC) key twice will place the cursor in the Format Window if the cursor was in the Format Window prior to Menu Bar activation.

If a pulldown menu is displayed, the (ESC) key must be pressed three times to access the Format Window if the cursor was in the Format Window prior to Menu Bar activation.

When the cursor jumps to the Format Window, it will always be located in its prior location in the Format Window.

Pointer Symbols Used with the COBOL FormPrint Format Window

The Format Window uses several different pointer symbols. The default pointer symbol, which is a diagonally pointing arrow, is used to position the cursor and select fields in the Format Window for moving, resizing or property modification.

The action pointer is used in the Format Window to locate a field or select a field in the Format Window for further definition. The action pointer is displayed as a result of clicking the mouse on the Icon Tool Bar.

The pointer will change to a double arrow symbol when a field has been selected and is displayed with a reverse video border, and when the pointer is directly on the reverse video border.

The double arrow symbol indicates that the mouse may be used to change the width and/or height of a field. There are three general types of double arrow symbols as follows:

Pointer Symbol	**Action to be Taken**
Horizontal Double Arrow	Change the width of the field
Vertical Double Arrow	Change the height of the field
Diagonal Double Arrow	Change the width and height of the field

Accessing the COBOL FormPrint Treeview Control

Using the mouse, position the pointer on the item which you want to establish as the current object. Double click the left mouse button.

CHAPTER B2
How to Use the COBOL FormPrint Form Editor

What is a Form File?

COBOL FormPrint stores form information in data files called form files. These form files contain the images of the forms you create as well as all of the information about the fields contained on each form.

COBOL FormPrint allows you to store one form per form file or many forms in a file. You can create any number of form files you desire. The form files may also contain full size form definitions as well as smaller form definitions.

These form files will eventually be used and loaded into memory by your COBOL application when you make a CALL to COBOL FormPrint. In order to design your form images, it is necessary to first create a form file. Once the form file is created, it may be opened later so that you may add additional forms or modify existing forms.

Before you begin designing your forms, you must first create a new form file or open an existing form file.

Create a New COBOL FormPrint Form File

To create a new form file select the File/New file Menu Bar Option.

The COBOL FormPrint Menu Bar may be accessed with the mouse by positioning the pointer on the desired menu option and clicking the left mouse button. If you prefer to use the keyboard to access the menu bar, press the (ALT) key or (F10) key to activate the menu bar. Use the (ARROW) keys to navigate the menu bar and pull down menus. When you have highlighted the New file pull down option, press the (ENTER) key.

The New File pop-up window will be displayed allowing you to establish the file name and directory in which the file will reside. There are no restrictions on file naming conventions for your form files other than those limitations imposed by your operating system/environment.

Don't change the default directory. This will allow you to create the form file in the directory in which the COBOL FormPrint Form Editor resides. Press the (ENTER) key or click on the OK Push Button to create the form file.

How to Use the COBOL FormPrint Format Window

The following will describe how to access the COBOL FormPrint Format Window and arrange fields to help you design your forms.

The Format Window is your work area. This is the area in which you arrange your fields for a form that will eventually be sent to your printing device from your COBOL application.

To scroll the format window, either click on the vertical and horizontal Scroll Bars or use the PGUP/PGDN keys to scroll up/down and the HOME/END keys to scroll left/right.

Working with Fields in the Format Window

Fields are easily created, sized and arranged in the Format Window with your mouse or the keyboard. When working with fields in the Format Window, avoid overlaying one field with another, unless you are creating a special effect by overlaying a static field with a data field or a bitmap field with a static field.

Creating a Field in the Format Window

Using the Mouse

Choose the data field icon (pictured above) from the Icon Toolbar and select it by positioning the pointer directly on the icon and clicking the left mouse button. Move the mouse pointer near the top left corner of the Format Window. This is the location in which the data field is to appear. Click the left mouse button once more. A default representation of the selected field will appear in the Format Window.

Using the Keyboard

When creating a field with the keyboard, the cursor should be positioned in the desired location within the Format Window before the field is created. Otherwise, the field will be automatically created in the current Format Window cursor position.

Once the cursor is properly positioned in the Format Window, press the (ALT) or (F10) key to activate the Menu Bar. Next, press the "d" mnemonic key or use the (ARROW) keys to display the Field Pull Down Menu. Position the highlight on the desired field type and press the (ENTER) key to create the field.

Selecting a Field in the Format Window

Using the Mouse

To select a field in the Format Window, double click the left mouse button while the pointer is directly on the field. The field will then be surrounded with a reverse video border, indicating that the field has been selected. If you wish to use the keyboard to select a field, first position the cursor on the field, then press the (ENTER) key to select it.

Using the Keyboard

To select a field in the Format Window with the keyboard, first position the cursor directly on the field.

Once the cursor is located on the field, press the (ENTER) key. The COBOL FormPrint Form Editor will then surround the field with a reverse video border indicating that the field has been selected.

De-selecting a Field in the Format Window

Using the Mouse

When using the mouse to de-select a field, click once on the selected field. You may also move the mouse outside of the Format Window to de-select the field.

Using the Keyboard

A field which has been selected in the Format Window may be de-selected by pressing the (ESC) key.

Changing the Size of a Field or Group in the Format Window

Using the Mouse

Select the data field with the mouse. This is accomplished by double clicking the left mouse button while the pointer is directly on the field. The COBOL FormPrint will then surround the field with a reverse video border, indicating that the field has been selected.

Move the mouse pointer to one of the sides or corners of the reverse video border surrounding the data field. As soon as the pointer changes to a double arrow symbol, you may change the height, width or both the height and width of the field. To change the width of the data field, position the pointer on the side of the selected data field until the pointer is a horizontal double arrow. Click and hold down the left mouse button.

Modify the width of the data field by moving the mouse pointer to the left or right while continuing to hold the left mouse button. When the desired data field width has been achieved, release the left button on the mouse.

Using the Keyboard

If you want to use the keyboard to change the size of fields in the Format Window, first locate the cursor directly on the field you wish to re-size with the (ARROW) keys.

Press the (ENTER) key to select the field. The field will be surrounded by a reverse video border, indicating that it was selected.

Changing the size of a field with the keyboard is a two step process. The first step is to press one of the four key combinations below, depending upon which side of the field you wish to change. When you press the key combination below, the "focus" changes to the appropriate side. You may then proceed to step two.

Key Combination	**Focus is on:**
(CTRL-UP ARROW)	Top of Field
(CTRL-DOWN ARROW)	Bottom of Field
(CTRL-RIGHT ARROW)	Right Side of Field
(CTRL-LEFT ARROW)	Left Side of Field

Step two allows you to actually increase or decrease the size of the field depending upon which key combination is pressed. The following table indicates the result based upon the focus and the key combination pressed.

Focus is on the Top of the Field

(CTRL-UP ARROW)	Increase the Height of the Field
(CTRL-DOWN ARROW)	Decrease the Height of the Field

Focus is on the Bottom of the Field

(CTRL-UP ARROW)	Decrease the Height of the Field
(CTRL-DOWN ARROW)	Increase the Height of the Field

Focus is on the Right Side of the Field

(CTRL-RIGHT ARROW)	Increase the Width of the Field
(CTRL-LEFT ARROW)	Decrease the Width of the Field

Focus is on the Left Side of the Field

(CTRL-RIGHT ARROW)	Decrease the Width of the Field
(CTRL-LEFT ARROW)	Increase the Width of the Field

Using the Properties Box

If you want to use the Properties Box to change the size of fields in the Format Window, first select the field which you want to modify and then set the width and height properties located in the COBOL FormPrint Properties Box.

> *PLEASE NOTE: Although it is possible to change the size of virtually every field, it may not be necessary in all cases. When the default caption for Static Text, Radio Buttons and Check Boxes is increased in size, the field will increase in width automatically.*
>
> *These fields will automatically increase in width to accommodate changes in the width of the caption or the width of the font used for the caption. This occurs when the caption exceeds the current field width. The fields will not automatically increase in height to accommodate a larger font style. If you wish to use a larger font, you must manually increase the height of the field as appropriate.*
>
> *In addition, a decrease in the length of the caption will not automatically decrease the width of the field. In some cases, the appearance of the form is improved if all fields in a field group (such as*

data fields) are equal in length. If the field width were automatically decreased, it would be impossible to have equal width fields with variable length captions.

Working with Cell Rows and Cell Columns

The Format Window of COBOL FormPrint contains a grid to help you change the location fields and change the size of fields appropriately.
Using the top reverse video border to change the size of a forms field may not produce the precise size that you may desire. After you have selected a field to change the size, you may wish to use the bottom border to change the height of a field.

The reason you may wish to use the bottom of the reverse video border to change the height is because COBOL FormPrint uses the top of the field to establish the default row position for the field. When you use the top border to change the size of the field, it will always "snap" itself to the top of the cell row. By using the bottom border to change the size, you will be able to change the field height more precisely.

Moving a field to a new location on the form is perfectly acceptable, because the top of the field will "snap" to a cell row in the grid. If you need finer control over the field position that the default grid allows, you may change this grid size at any time using the Cell width and Cell height properties in the Properties Box.

Moving a Field in the Format Window

Using the Mouse

Select the data field on the Format Window by double clicking the left mouse button while the pointer is directly on the data field. The Form Editor will then surround the field with a reverse video border, indicating that the field has been selected.

Move the mouse pointer inside the reverse video border surrounding the field, then press and hold down the left mouse button. The mouse pointer must be inside the border in order to move the field. If the pointer is directly on the reverse video border, the field will be re-sized instead of moved.

Change the location of the data field by moving the mouse pointer while continuing to hold the left mouse button. The reverse video border will follow the pointer as long as the left mouse button is depressed. When the reverse video border for the field has been moved to the desired location, release the left mouse button. This process results in "dragging" the field to the desired location.

Using the Keyboard

If you want to use the keyboard to move fields on the screen, first locate the cursor directly on the field you wish to move with the (ARROW) keys.

Press the (ENTER) key to select the field. The field will be surrounded by a reverse video border, indicating that it was selected.

Move the field to the desired location by pressing the appropriate (ARROW) key.

Using the Properties Box

If you want to use the Properties Box to change the location of fields in the Format Window, first select the desired field and then modify the location by changing the values contained in the Row and Column properties in the Properties Box.

Deleting a Field in the Format Window

Using the Mouse

Move the mouse pointer to the "Delete" icon on the Icon Toolbar and click the left button once. The pointer symbol will change to an action pointer. Move the pointer so that it is directly on the data field and click the left mouse button once more. The data field will be deleted.

Using the Keyboard

Move the cursor to the field that you want to delete in the Format Window. Select the Field/Delete Menu Bar Option. The field will be automatically deleted. You should use caution when selecting a field to be deleted. When this method is used, there is no warning given before the field is deleted.

Manipulating Multiple Fields and Groups in the Format Window

You may find it useful to move a whole area within the format window rather than individually moving the fields and/or groups within that area. You can also copy fields and/or groups within the window to the clipboard and later copy those items back to a different place within the same panel or even to another panel. In order to manipulate multiple fields and groups, you must first establish a selection area which includes the fields and/or groups you want to manipulate.

Selecting an area using the mouse

An area of the form is selected by holding down the left mouse button and moving the mouse pointer down or to the right. After the mouse pointer has been moved a few pixels, the selection highlight will appear and will resize itself as the mouse pointer is moved further. When the required area has been highlighted, release the mouse button.

Selecting an area using the keyboard

Alternatively, you may position the cursor using the arrow keys and then use the CTRL-RIGHT and CTRL-DOWN keys to select an area.

Moving the selected area using the mouse

To move the items included in the selected area, move the mouse pointer into the selected area, hold down the left mouse button, move the selection highlight to a new location and then release the mouse button.

Moving the selected area using the keyboard

Alternatively, you may use the arrow keys to move the selection highlight and then press ENTER.

Copying the selected area

Once an area has been selected, the items within it may be copied to the editor clipboard using the Edit/Copy menu option.

Deleting the selected area

Once an area has been selected, the items within it may be deleted using the Edit/Clear menu option.

Deleting and copying the selected area

Once an area has been selected, the items within it may be copied to the editor clipboard and then deleted in one action using the Edit/Cut menu option.

Copying an area back to the Format Window

Once items have been copied to the editor clipboard using the Edit/Copy or the Edit/Cut menu options, these items may be copied back to the current form or a different form using the Edit/Paste menu option. To do this, position the cursor using the mouse or arrow keys and select the Edit/Paste option. The items will be copied to the area whose top left corner is defined by the cursor position. The Paste option will maintain all the IDs of selected items unless those IDs are already being used in the form. This may occur if items are being moved to another form or copied within the same form. The Paste option will also adjust the size of items if items are being moved to a form with a different cell size. The contents of the editor clipboard will be maintained (for use in Paste operations) until another Copy or Cut operation is performed or a selected area is moved (see Moving the Selected Area above).

Overlaying One Field with Another in the Format Window

When creating or moving fields in the Format Window, it is possible to overlay one field with another. When a field is overlaid with another in the Format Window, it may cause the overlaid field to disappear from the Format Window. To restore the field back to its original state, move the top field.

Undoing Your Last Change

If during editing, you make a change to the form in error or a change that you are not satisfied with, you can undo the change by selecting the Edit/Undo menu option. Any edit action is considered a change, including adding and deleting fields, moving fields and changing attributes by selecting menu options. Only the last edit action can be undone.

Aligning Fields

Once an area has been selected, the fields within that area may be aligned using the Edit/Align menu option. Fields can be aligned with respect to the first field in the selected area either horizontally (left/right/center) or vertically (top). Fields can also be resized to the first field and spaced horizontally or vertically by a specified amount.

CHAPTER B3
Working with Bitmaps in the Form Header

The first part of the sample form to be designed will be the header. The header in the sample form will contain a bitmap for the company logo, a form title and fields for the page number and date that the form is printed.

Using a Bitmap to Print the Company Logo on a Form

Because the Bitmap Field will be established in the top left area of the form, you should scroll the form image up and to the left to view the top left corner of the form. To scroll up, click the mouse on the upward pointing arrow on the vertical scroll bar. To scroll left, click the mouse on the left pointing arrow on the horizontal scroll bar. The form will shift to reveal the top left portion. If you are not using a mouse to design your forms, please refer to the instructions in Chapter B2, How to Use the COBOL FormPrint Form Editor for information on scrolling through the form with the keyboard.

What is a Bitmap?

A bitmap is a raster based (pixel) graphical image. Bitmaps are useful to display graphics on your form. Because bitmaps are images, the graphic image in your bitmap is only limited by your imagination. One example of a useful bitmap is your company logo as a header on your form. Another example might be a picture of an inventory item or an employee photograph.

Bitmaps should be in DIB (Device Independent Bitmap) format and have the extension ".BMP". COBOL FormPrint can also support compressed JPG (JPEG) format bitmap files with a special add-on tool. Please contact your distribution agent for more information on the JPG alternate bitmap image add-on.

Add a Bitmap to the Format Window

To create a bitmap field, click the left mouse button on the "Bitmap" icon or select the Field/Bitmap Menu Bar Option.

If you are using a mouse, move the pointer near the top and left of the Format Window and click once. The bitmap you are adding will simply be a logo that will be added to the form in order to improve its appearance.

Select the Value Property in the Properties Box. A small pop-up box containing three entry fields will be displayed. The second field in the pop-up box has a caption of "Icon file." This is the property in which you will type the name of the bitmap image file which you wish to use.

If you aren't sure of the name of the bitmap image which you plan to use for your bitmap field, click the small downward pointing arrow at the far right side of the "Icon file" entry field in the pop-up box. This will display the Windows File Open dialog box which will allow you to locate and select the file name from a list of files.

You may now type the file name, "FORMPRNT.BMP" directly into the File Name property or scroll through the list box on the Open pop-up window until you highlight the file name. If you choose to select the file name with the highlight, press the (ENTER) key when the proper file name has been highlighted.

You will now see that the Value Property in the Properties Box contains the file name, "FORMPRNT.BMP." If you already know the name of the bitmap file which you want to include in the form, you may also type it directly into the Value Property in the Properties Box.

You should also notice that the Type Property in the Properties Box displays x"00" which is a regular bitmap image. If you want to print 256 color bitmaps from your COBOL FormPrint form, then you may wish to establish the icon field as a "bitmap exact" in the Type Property. This has the effect of printing the exact 256 color palette for the bitmap. Please bear in mind that each 256

color bitmap may have a unique palette, therefore you may get unpredictable results if you attempt to print multiple 256 color bitmap icons on the form simultaneously.

The bitmap image that you have added to the form is larger than the default bitmap size, so it is necessary to resize the field to fit the size of the bitmap. This is accomplished with the mouse or the keyboard using the same method as resizing any other field in the Format Window. The easiest way to accomplish the desired result is to allow COBOL FormPrint to automatically resize the bitmap image.

Set the Special format property to "i = size field to image." See Chapter B1 for details on setting properties.

If you prefer to change the size of the bitmap field manually, you may do so as follows:

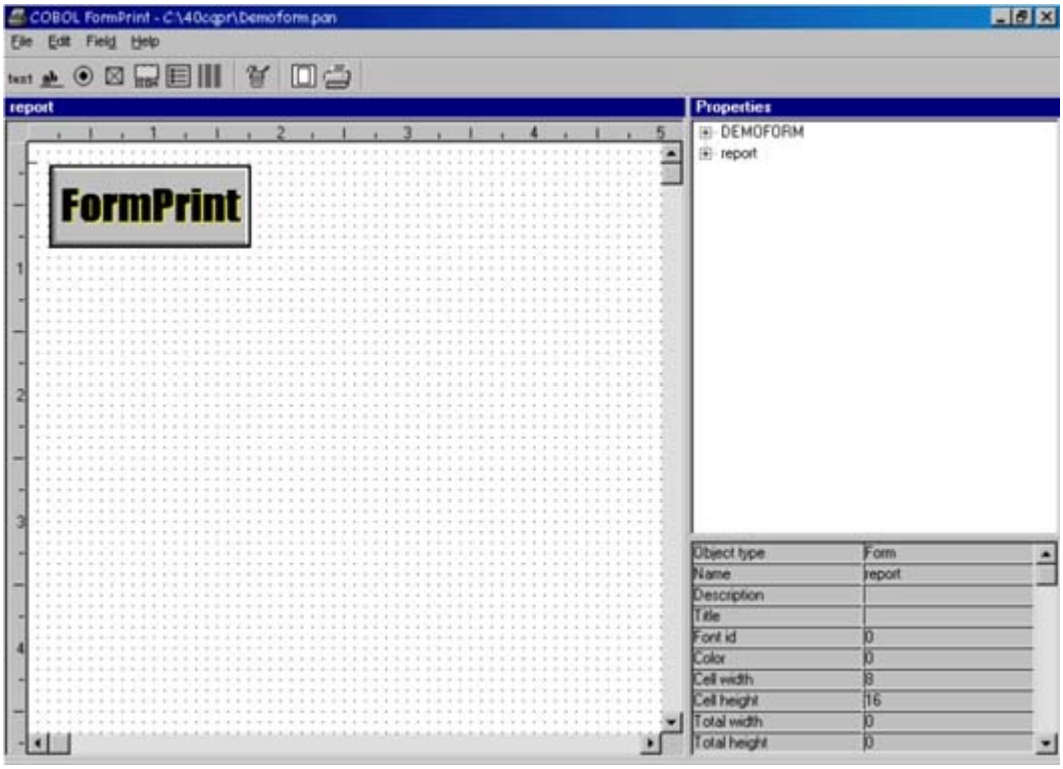
Change the size of the bitmap by double clicking the left mouse button on the field in the Format Window. You may also select the bitmap field with the (ENTER) key.

A reverse video border will appear around the graphic. Move the pointer to the bottom right corner until the pointer changes to a diagonally pointing double arrow. Click and hold the left mouse button.
Drag the pointer to the right and down to change the size of the field. Release the left button to establish the size of the field. If the new size is still too small or too large for the graphic, you may repeat this procedure until the field and bitmap are the same size.

If you are using the keyboard to re-size the bitmap field, use the (CTRL-ARROW) keys to increase or decrease the size of the bitmap field once it has been selected.

After you have changed the size of the bitmap field, move the bitmap image to the top left corner of the form. Moving a bitmap field is accomplished in the same manner as moving any field type in COBOL FormPrint. For instructions on moving a field, please refer to Chapter B2, How to Use the COBOL FormPrint Form Editor.

Your form in the Form Editor should now look like this:



CHAPTER B4
Working with Static Text in the Form

What is Static Text?

Text

In COBOL FormPrint, static text is generally used to represent a field label. This is how it will be used in this example. Static text could also be used anytime you want to add information to the form which normally wouldn't change.

Create a Line of Static Text in the Form Definition

Using the Mouse

Click the mouse pointer on the "Static Text" Icon. The pointer will change from a diagonal arrow to a vertical arrow, indicating that the icon has been selected. Move the pointer to the center of the Format Window and click once to establish the static text. The default value is "Text."

Using the Keyboard

You may also select the Field/Static text Menu Bar Option to create Static Text in the Format Window. You should use caution when creating fields using the COBOL FormPrint Menu Bar. The field will be automatically positioned where the cursor is located in the Format Window. It is important to first locate the cursor to the desired field location in the Format Window before activating the Menu Bar to create the field.

Move the pointer to the Text Property in the Properties Box and click the left mouse button. This is the property in which you will establish the actual text for the static text. Type "Inventory Report" and delete the extra text characters at the end of the field using the (DEL) key. Press the (ENTER) key. You will see that the text in the Format Window has been changed to the new value.

Important Issues Involving Static Text in COBOL FormPrint

It is important that all static text fields used in your COBOL FormPrint form definitions use a special font other than the system default font. Changing the font style in your form definition is covered in the next chapter. In addition, any fonts used to develop your form definitions also must exist in the target environment where the printing will take place. This is necessary, because FormPrint needs to have the specific True Type Font in order to print the typeface correctly on the target environment. It is generally a good idea to try to limit the fonts you use to design your form to those fonts which are distributed with the standard Windows environment.

These fonts which are always present in a Windows system include:

Courier New
Times New Roman
Arial

PLEASE NOTE: Do not use raster based fonts such as Bitstream fonts. Unpredictable results can occur. You should always use Microsoft Windows True Type Fonts.

Multiline Text

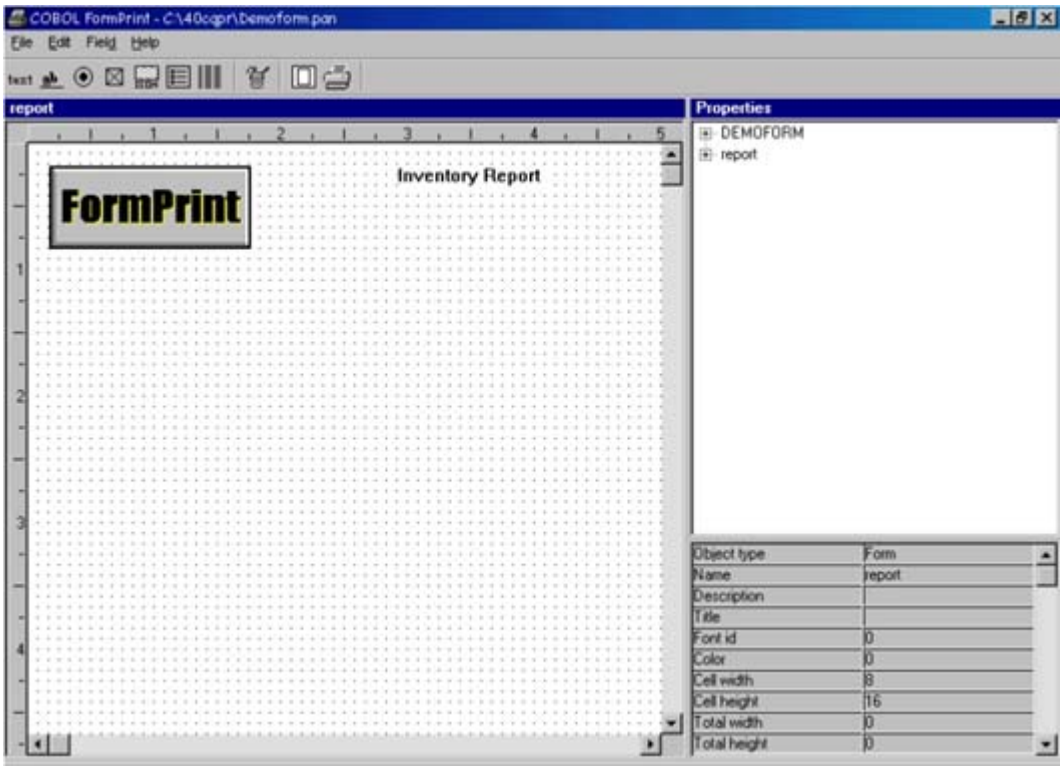
Static text can be spread over multiple lines by setting the Type Property to the value of "m = multi-line text." This will cause the text you type in to wrap within the area you have allocated.

Move the Line of Static Text in the Form Definition

If you are not using a mouse to design your forms, please refer to the instructions, in Chapter B2, How to Use the COBOL FormPrint Form Editor, for information on moving and re-sizing fields with the keyboard.

The only problem with our line of static text is that it should be repositioned to the top center of the Inventory Report form. Reposition the text by double clicking on the static text in the Format Window. A reverse video border will appear around the text. Move the pointer to the center of the static text, click once and hold the left mouse button down. Drag the mouse pointer up until the static text is at the top and center of the Format Window. Release the left mouse button to establish the new location.

The Inventory Report form should now look like this:



2. Save the form by selecting the Save pull down option from the File menu bar option.

CHAPTER B5
Changing the Default Font Size and Type

What is a Font?

A Font is a specific style of typeface used on the form. Fonts can have a number of attributes including:

- The typeface name, representing the style of the font, such as Times New Roman or Arial. This text is displayed using the Times New Roman font. This text is displayed using the Arial font.
- The size of the font, expressed in terms of point size or width and height. This font size is a 10 point size. This font is a 12 point size.
- The font pitch which can be a fixed pitch (or monospace) or a proportional pitch (variable) pitch. All characters in a fixed pitch font use the same amount of horizontal space on the form. Characters in a variable pitch font vary in width. For example, in a variable pitch font, an "i" requires much less horizontal spacing than an "m."
  Variable pitch font: mmiimmiimm
  Fixed pitch font: mmiimmiimm
- The weight of the font refers to the amount of emphasis which is placed on the typeface. Fonts may be normal or bold. Bold fonts are generally darker and thicker. This is a regular weight font. **This is a bold weight font.**
- Other font attributes can include: italics, strikethrough and underline.
  This font is in italics.
  ~~This font is in strikethrough.~~
  This font is in underline.

All static text fields must have a font defined. Only True Type Fonts can be used.

Important Issues Involving Fonts in COBOL FormPrint

It is important that the fonts used to develop your form definitions also exist in the target environment where the printing will take place. This is necessary, because FormPrint needs to have the specific True Type Font in order to print the typeface correctly on the target environment. It is generally a good idea to try to limit the fonts you use to design your form to those fonts which are distributed with the standard Windows environment. These fonts which are always present in a Windows system include:

Courier New
Times New Roman
Arial

> *PLEASE NOTE: Do not use raster based fonts such as Bitstream fonts. Unpredictable results can occur.*

Add Static Text to the Form

If you have not already done so, add the static text, “Inventory Report” in the top center of the form. If you are not sure how to add a line of Static Text to the form, please refer to Chapter B4, Working with Static Text in the Form.

Add a New Font Size and Type

Select the Font id Property in the Properties Box.

The Fonts pop-up window will be displayed. The two default fonts are both the system default fonts. Because the text will be changed to a new typeface, position the pointer on the Fonts drop down list and click the right mouse button. A small pop-up menu will be displayed allowing you to establish a new font or preview an existing font. Select the New menu option to establish a new font.

The "Font" pop-up window will be displayed to allow you to specify a new font. Add the following to the Font pop-up window:

Font:	Times New Roman
Font style:	Select “Bold” weight
Size (in points):	20
Effects:	use the default settings

Press the (ENTER) key or click the "Ok" Push Button to return to the Font selection pop-up window.

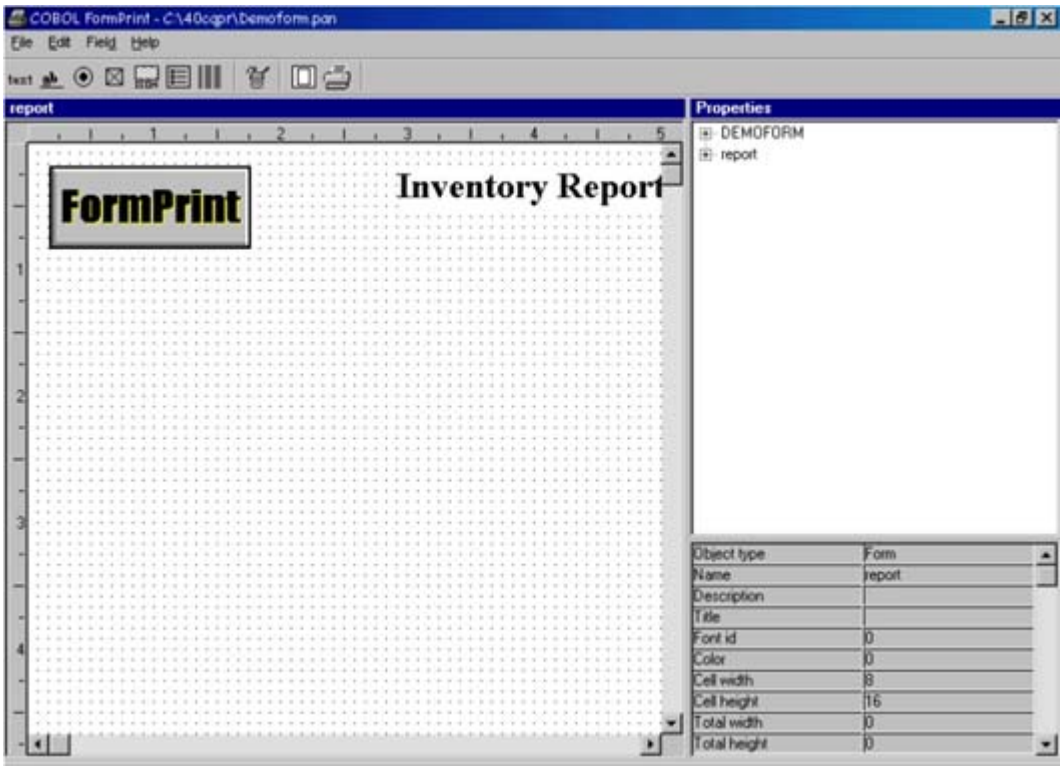
Change the Default Font

You will now see that the new font has been added to the drop down list of available fonts for your form. Highlight the "Times New Roman" font with a width of 12 pixels, a height of 31 pixels, variable pitch and bold weight and press the (ENTER) key. Press the (ESC) key to return to the Format Window.

The Static Text is now larger than the original text. This means that you must increase the size of the Static Text field.

Increase the size of the Static Text in the same manner as you increase the size of any field.

The Inventory Report form should look like this now:



PLEASE NOTE: Only True Type Fonts can be used with COBOL FormPrint. Remember that any fonts which you use on your forms in your development environment must also exist in the target environment where the printing will take place. That means if you use a True Type font which is unique or one which is not distributed with a standard Windows environment, you must make arrangements to have that font installed on the target environment.

CHAPTER B6
Changing Colors

Establish a Color for a Form Field

If you have not already done so, add the static text, "Inventory Report" in the top center of the form. If you are not sure how to add a line of Static Text to the form, please refer to Chapter B4, Working with Static Text in the Form. To change the color for this field, first make sure that the cursor is on the Static Text field and then select the Color Property in the Properties Box.

The Colors drop down list will be displayed.

The two default colors are both the system default colors. Because the color will be changed to a new color, position the pointer on the Colors drop down list and click the right mouse button. A small pop-up menu will be displayed allowing you to establish a new color or preview an existing color. Select the New menu option to establish a new color.

The Color definition pop-up window will display to allow you to specify a new color

To establish a new color scheme for the static text field, you must first establish a new color set. The colors used for fields in COBOL FormPrint can be regular colors or special RGB colors.

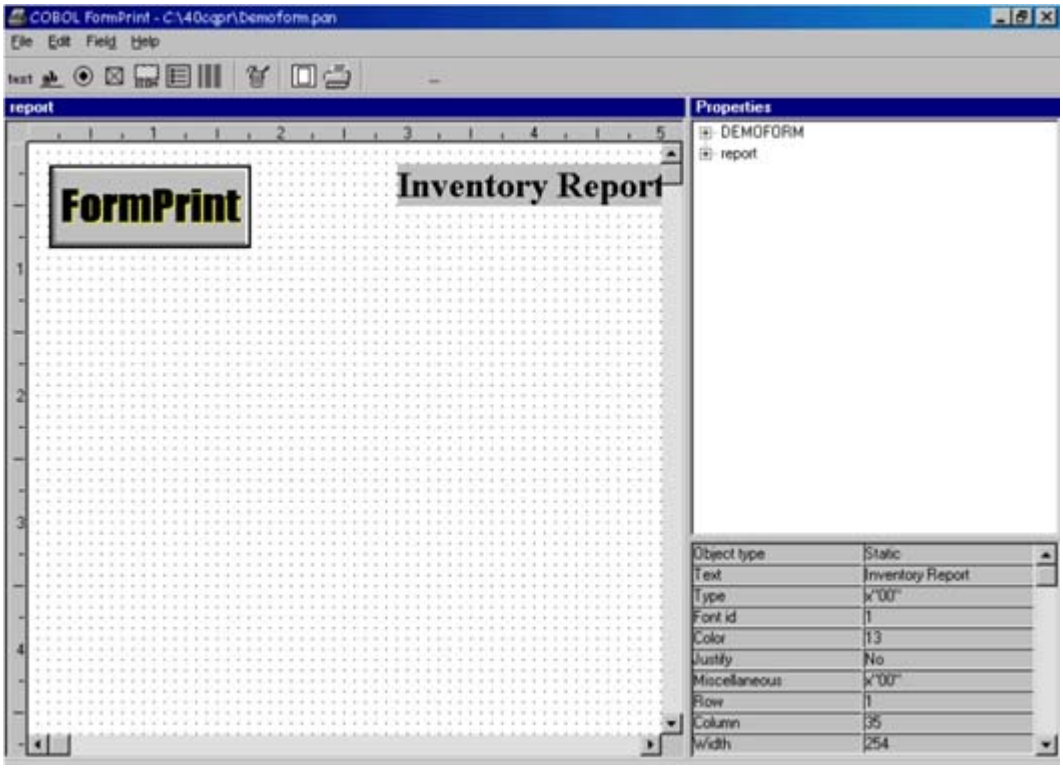
To select an RGB color for the foreground color, click once on the combination box field under the word, "foreground." Scroll down the drop down list until you see the option entitled "RGB-specific." Select the "RGB-specific" foreground color option by clicking on it once. Now click on the pushbutton with the caption of "RGB." You will see the color pop-up display. If you click on the "define custom colors" pushbutton, you can specify a very specific RGB color, but for now, just select a the color in the first column, sixth row, which is the bottom left color (should be black). Press the (ENTER) key or click on the Ok pushbutton.

Next do the same procedure for the background color. Select an RGB color of light gray which is the color block on the sixth column and sixth row.

Click on the "Ok" Push Button or press the (ENTER) key to add the new color combination to the list of available colors in your color list. The new color you have just added will be the last color listed in the color list. It will also show as a an R-000 G-000 B-000 foreground color (black) and an R-192 G-192 B-192 background color (light gray). Make sure that the new color is highlighted and select it with the mouse or press the (ENTER) key. You will see that the static text field color has now been changed to black text on a light gray background..

Click the mouse in the Format Window or press the (ESC) key to return to the Format Window.

Your "Inventory Report" form should now look like this:



6. Save the form definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

PLEASE NOTE: Although you can use any combination of RGB foreground and background colors in FormPrint, you should be careful in choosing colors that will print properly on the target printing devices.

For example: Although a black foreground on a red background may appear to be acceptable on a color printer, on a gray scale printer, such as a laser printer, the red background will print a very dark color. This may make the form unreadable. Try to use color combinations which offer a high degree of contrast between the foreground and background RGB colors.

CHAPTER B7
Working with Custom Fields

What is a Custom Field?

A Custom Data Entry Field is a field on the form into which you can move data from your COBOL application.

Custom Fields are useful for creating single fields or for creating Repeat Groups. The next few chapters will demonstrate this type of complex form field.

Add Several Custom Fields to the Form

Using the "Custom Data Entry Field" icon or the Field/Custom field Menu bar Option, add a Custom Field to the right of the "Inventory Report" static text field. Make sure you leave enough room between "Inventory Report" and the Custom Field to place a static text field label. Increase the width of the Custom Field by 10 cell columns. This can be easily accomplished by changing the Width Property in the Properties Box from its default value of "50" to a value of "150."

Make sure that the cursor is still located on the Custom Field you just created and add a COBOL Field Name to the field by typing the following text into the Name Property in the Properties Box:

PAGE-NUMBER

If you prefer to use the keyboard, make sure that the cursor is on the Custom Field and press the (ESC) key to position the cursor inside the Properties Box. Position the cursor on the second property field in the Properties Box. This is the Name Property.

Add a Static Text Field to the immediate left of the Custom Field and change the Text Property from the default value of "Text" to "Page:."

Add a second Custom Field directly below the first and increase its size to the same width as the first (Width Property = 150). Add the following COBOL Field Name to the Name Property:

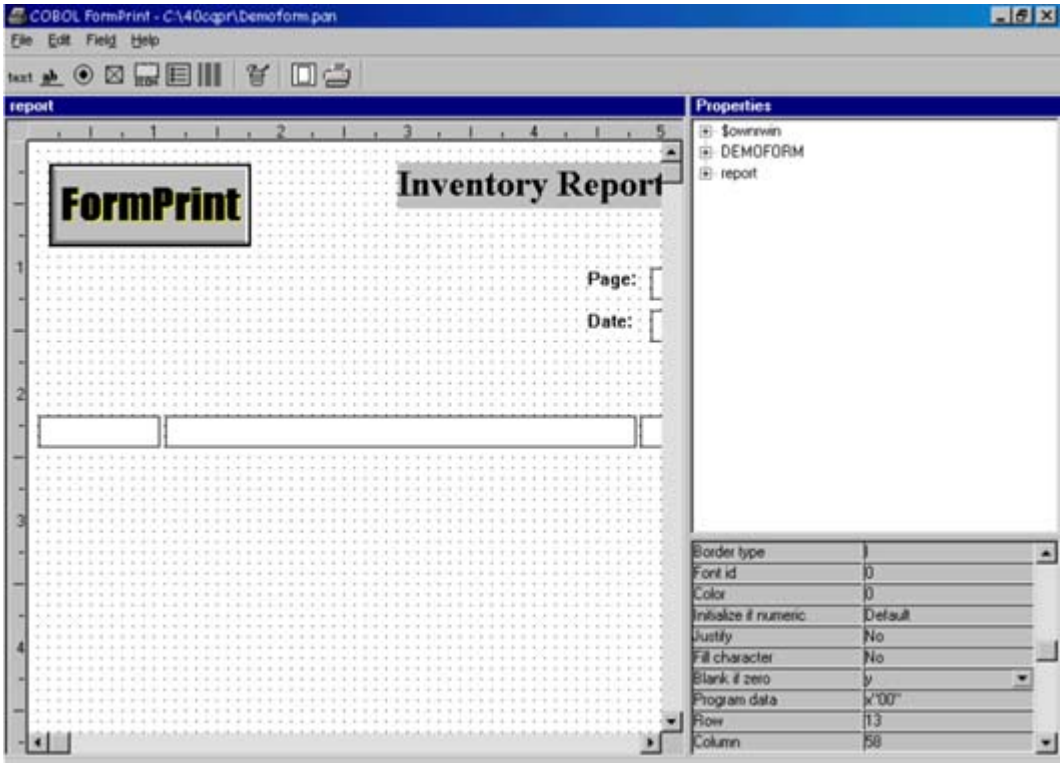
REPORT-DATE

Add a Static Text Field to the immediate left of the field and change the Text Property from the default value of "Text" to "Date:."

Next, create the following four fields in a horizontal arrangement, at least 6 cell rows below the form heading. Start the first field below the "FormPrint" bitmap logo and locate the remaining fields to the right of the first in sequential order. Increase the size of each field according to the following specifications:

Field Name	**Width Property**
ITEM-NUMBER	55
DESCRIPTION	505
QUANTITY	55
PRICE	100

Your Inventory Report Form should now look like this:



Multiline Fields

Field data (text) can be spread over multiple lines by setting the Usage option Property in the Properties Box to a value of "m = multi-line." This will cause the text you pass from your program to wrap within the area you have allocated to the field. Remember to set the COBOL field format property large enough to accommodate the text (see the next chapter for details on setting up this format).

CHAPTER B8
Working with Input Types

What is an Input Field Type?

An Input Field Type represents the type of allowable data which may be passed to a Custom Data Field. The Input Field Type may be Alphanumeric, Numeric or Date.

A Field Type of Alphanumeric (any data) will allow passing of any keyboard character as the name implies. A Numeric Field Type will only allow passing of numeric data and a Date Field Type will allow passing of a valid date.

Once the Input Field Type is set for the Type Property, it is still necessary to establish an appropriate COBOL Field Format in the Format Property in the Properties Box.

Establish a Numeric Field

Place the cursor on the Page Number Field by clicking the mouse on the field or using the (ARROW) Keys to position the cursor.

Set the Format Property in the Properties Box to 9(3). If you prefer to use the keyboard, press the (ESC) key once and then use the (ARROW) keys to move the cursor to the Format Property field.

Establish the Input Type

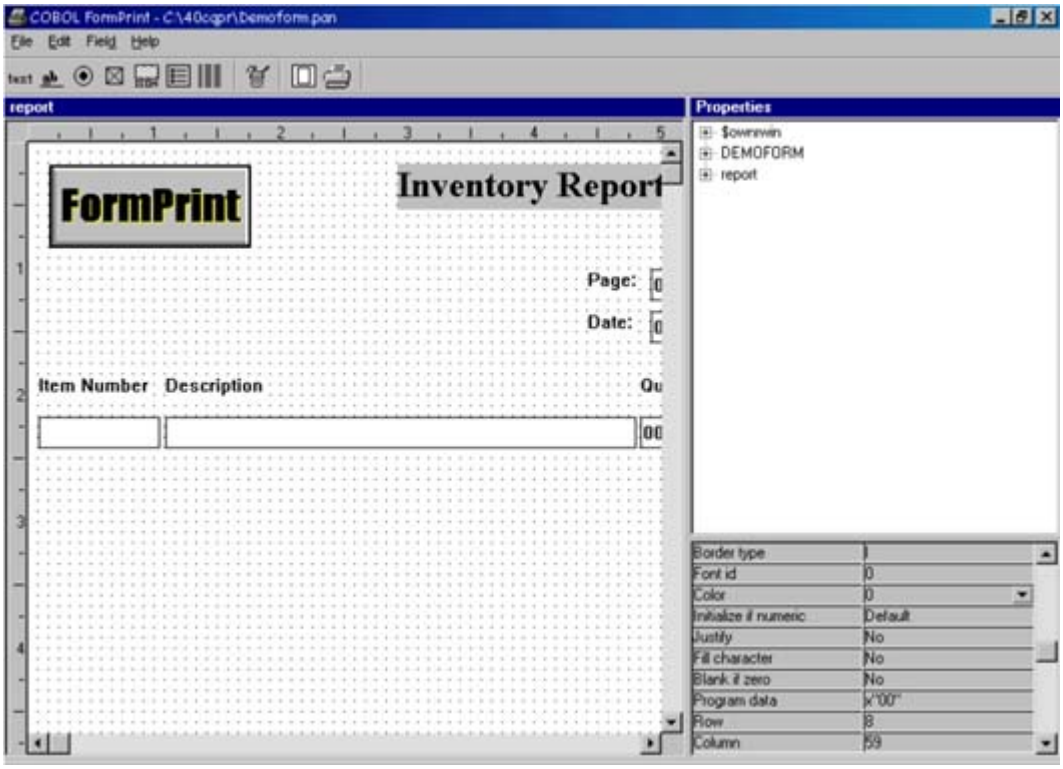
Now set the Type Property to “n = numeric.” Press the (ENTER) key and then the (ESC) key to return to the Format Window, or click the mouse once in the Format Window to return.

If you prefer to use the keyboard, first make sure that the cursor is located directly on the Page Number field. Press the (ESC) key to activate the Properties Box. Use the (ARROW) keys to position the cursor in the Type Property Field. Press the (ENTER) key to display the drop down list for the Type Property. When the list of options for the input type is displayed, use the (DOWN ARROW) key to highlight the "Numeric" option and press the (ENTER) key to select the option.

Now modify the COBOL Field Formats and Field Input Types for the other fields on the Inventory Report Form as follows:

Field Name	Format Property	Type Property
REPORT-DATE	MM/DD/YYYY	Date
ITEM-NUMBER	X(12)	Alphanumeric
DESCRIPTION	X(25)	Alphanumeric
QUANTITY	9(5)	Numeric
PRICE	$(5).99	Numeric

Your Inventory Report Form should now look like this:

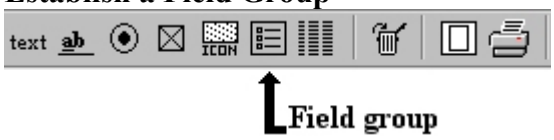


CHAPTER B9
Working with Field Groups

What is a Field Group?

A field group is used to draw a box around fields. The interior of this box may also be filled with an RGB color.

Establish a Field Group



Now that you have a number of custom data fields on your form, you should establish a field group. To accomplish this, position the pointer on the group icon and click the left mouse button once. Move the pointer to the top right corner of the Static Text for the Page Number Field in the Format Window. Click once to establish the group box.

Using the Keyboard

You may also select the Field/Group box Menu Bar Option to create a Field Group in the Format Window. You should use caution when creating Groups using the Menu Bar. The Group will be automatically positioned where the cursor is located in the Format Window. It is important to first locate the cursor to the left and immediately above the first field to be included in the Group before activating the Menu Bar to create the field.

Include All Fields Within the Field Group

A box with a reverse video border will appear. If the box is a thin, line, the Group Box was "de-selected." To change the size of the Group Box, select it by double clicking the mouse on the thin line until it becomes a thick reverse video line. You may also select the Group with the keyboard by first positioning the cursor on the Group and pressing the (ENTER) key.

To change the size of the group move the pointer to the lower right corner of the reverse video border until the pointer changes to a diagonally pointing double arrow symbol. When the pointer becomes a double arrow, you can resize the reverse video border down and to the right, until the Group Border includes the Static Text captions as well as the Custom Data Fields for the Page Number and the Report Date Fields.

Release the left mouse button to establish the Field Group. The reverse video border will become a thin reverse video box around the fields in the group. If the box doesn't include all the fields in the group, you may delete the field group and start over or change the size to include all the fields in the group. For more information on deleting a field, please refer to Chapter B2, How to Use the Form Editor.

Using the Keyboard

If you want to use the keyboard to change the size of the Group in the Format Window, first position the cursor directly on the Group you created with the (ARROW) keys.

If the Group isn't selected, press the (ENTER) key to select the field. The field will be surrounded by a reverse video border, indicating that it has been selected.

Change the size of the Group by pressing the (CTRL-RIGHT ARROW) key combination to increase the width and (CTRL-DOWN ARROW) key combination to increase the height.

Using the Properties Box

If you want to use the Properties Box to change the size of the Group in the Format Window, first position the cursor directly on the Group you created and then set the desired Group box size by modifying the Width and Height Properties in the Properties Box.

Establish a Fill Color for the Field Group

To change the fill color for the group, select the Color Property in the Properties Box. A drop down list of available color selections will be displayed. Scroll down the drop down list until you see the color combination with a foreground color of R-000 G-000 B-000 and a background color of R-192 G-192 B-192. This is an RGB color of a black foreground on a light gray background. Select this color combination by double clicking it with the mouse or you can highlight the desired color combination and press the (ENTER) key.

If your drop down list does not display this specific color combination, please refer to Chapter B^, Changing Colors, to learn how to create a new color combination.

Change the color for the static text “Page” and “Date” included within the group box. Use the same color combination as you did when filling the group box with the black foreground and light gray background color.

Also change the color for the two custom fields contained within the group box.

Create Static Text Column Headings in the Form Definition

Click the mouse pointer on the "Static Text" Icon. The pointer will change from a diagonal arrow to a vertical arrow, indicating that the icon has been selected. Move the pointer approximately 3 cell rows above the ITEM-NUMBER field and click once to establish the static text. The default value is "Text."

Using the Keyboard

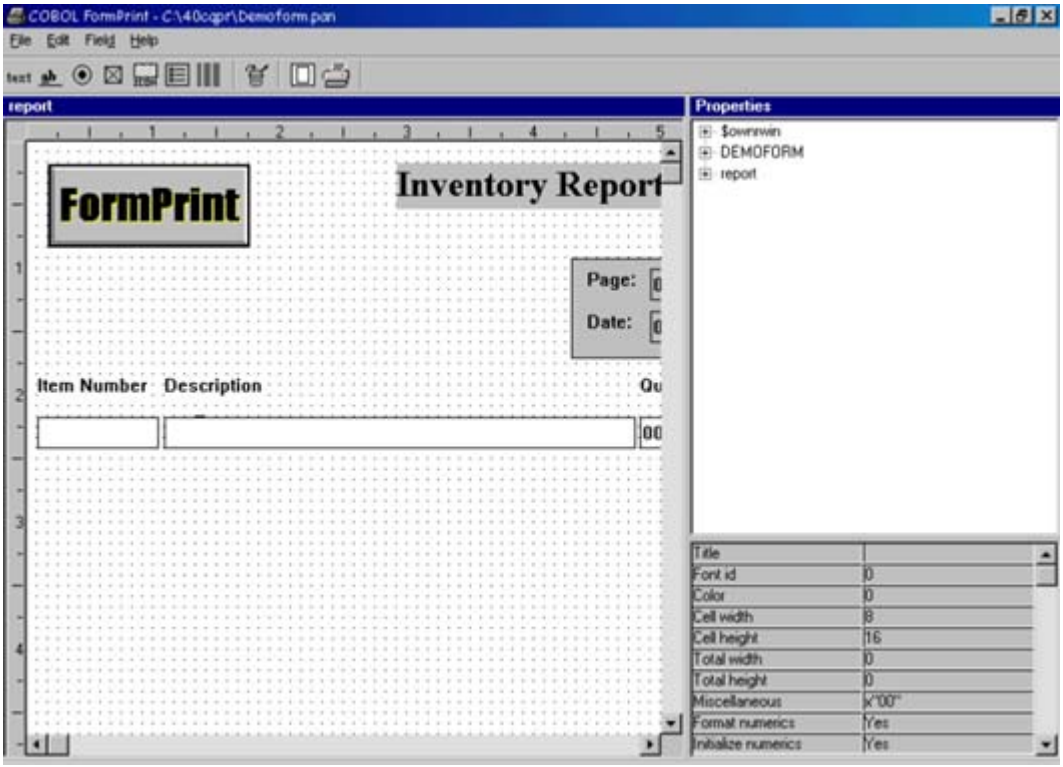
You may also select the Field/Static text Menu Bar Option to create Static Text in the Format Window. You should use caution when creating fields using the COBOL FormPrint Menu Bar. The field will be automatically positioned where the cursor is located in the Format Window. It is important to first locate the cursor to the desired field location in the Format Window before activating the Menu Bar to create the field.

Move the pointer to the Text Property in the Properties Box, type "Item Number" and delete the extra text characters at the end of the field using the (DEL) key. Press the (ENTER) key. You will see that the text in the Format Window has been changed to the new value.

Using the same procedure, add three more static text column headings above the following fields:

DESCRIPTION
QUANTITY
PRICE

Your Inventory Report Form should now look like this:



PLEASE NOTE:

Although you can use any combination of foreground and background colors, you should be careful in choosing colors that will print properly on the target printing devices.

For example: Although a black foreground on a red background may appear to be acceptable on a color printer, on a gray scale printer, such as a laser, the red background will print a very dark color. This may make the form unreadable. Try to use color combinations which offer a high degree of contrast between the foreground and background colors.

CHAPTER B10
Working with Repeat Groups

What is a Repeat Group?

A Repeat Group is a group of one or more Custom Data Fields with multiple vertical occurrences on the form.

Add a Repeat Group on the Panel

If you are not using a mouse to design your forms, please refer to the instructions, in Chapter B2, How to Use the Form Editor for information on establishing a Repeat Group with the keyboard.

Select the "Repeat Group" icon to repeat the Custom Fields on the form. Position the mouse pointer in the top left inside corner of the “Item Number” Custom Field. Click the left button once.

Change the size of the Repeat Field Box so that it extends to the right and below all four the Custom Fields in the detail area of your form. This should include the Item Number, Description, Quantity and Price Custom Data Fields.

Now select the Repeat Field Group Box again by double clicking the left mouse button while the pointer is on the Repeat Field Group Border. Move the pointer to the bottom dark highlight border of the selected Repeat Field Group Box.

When the pointer symbol changes to a double arrow symbol, click and hold the left mouse button and drag the mouse downward. This will result in increasing the height of the Repeat Field Group. Increase the height of the Repeat Field Group until you reach the bottom of the form definition. You will be at the bottom of the form when the form will not scroll downward any longer. You can also watch the slider box on the vertical scroll bar. When the slider box reaches the bottom of the vertical slider bar, you have reached the bottom of the form.

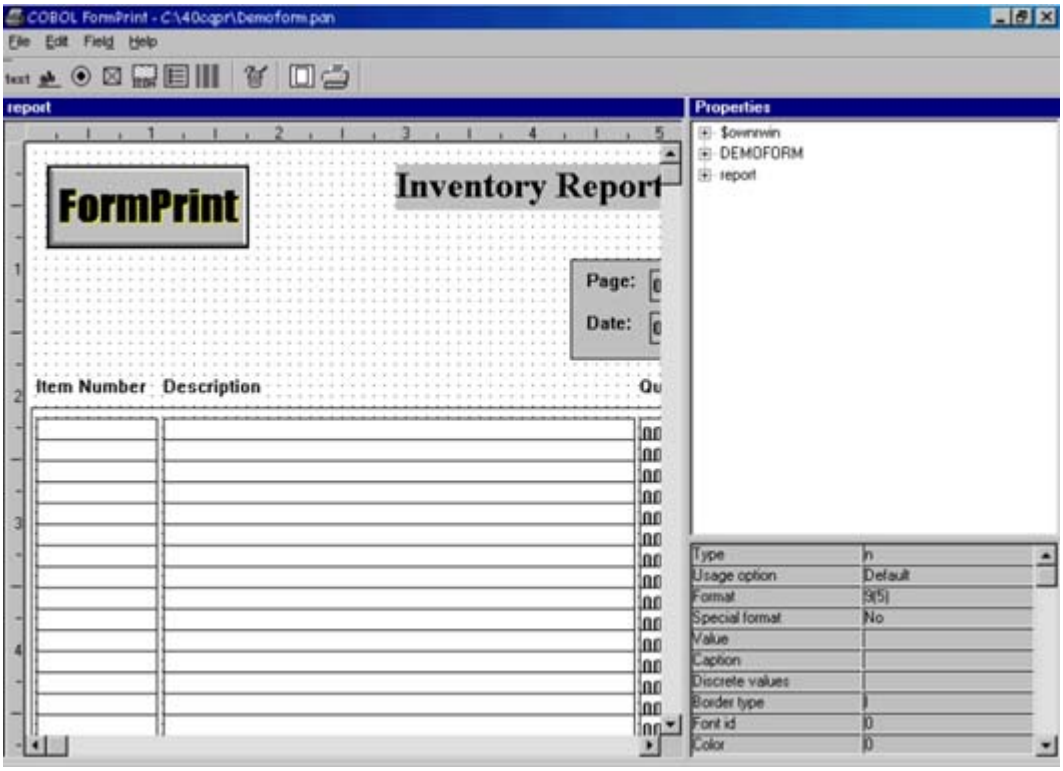
Specify the Total Number of Field Occurrences

With the cursor in the Format Window on the Repeat Field Box, select the Vertical occurs Property. This is where you will establish the number of times the row of fields in the Repeat Group will be repeated. If the Vertical occurs Property isn’t displayed in the Properties Box, the cursor is not on the Repeat Field Box.

Change the Vertical occurs Property from "1" to "50" and press the (ENTER) key. You should see the Custom Fields repeat down the form within the Repeat Group Box. You should also scroll down to the bottom of the form definition to make sure that all the repeated fields are visible on the form definition. If a field is not visible on the form definition, it will not be printed on the form. If you’re not sure whether or not all the repeated fields are visible, reduce the number of occurrences to 40 or 45. You should see a difference in the number of repeated rows if you have designed a form that is a standard A4 or 8.5” x 11” paper size.

After you have repeated the fields, you should notice that it appears that the field borders are overlapping one another. This doesn’t cause any problems for our form, because the borders will be removed. The next chapter will cover the issue of field borders.

Your Inventory Report form should look like this now:



CHAPTER B11
Changing the Field Border

What is the Field Border?

A Field Border is the thin line which surrounds various form fields in COBOL FormPrint. For example, both a Custom Data Entry Field and a Field Group contain a line which surrounds the data that is displayed in these fields.

COBOL FormPrint allows you to keep the default border, replace the default border with a 3-dimensional border or remove the border from the field.

Form fields which allow modification of the border include:

- Custom Data Fields
- Field Groups
- Repeat Groups

Eliminate the Border for a Field

The Field Borders in the Inventory Report form should be removed. For these types of field, the border is not necessary. Position the mouse pointer on the Page Number Custom Data Entry Field in the Repeat Field Group and click once to move the cursor to the field.

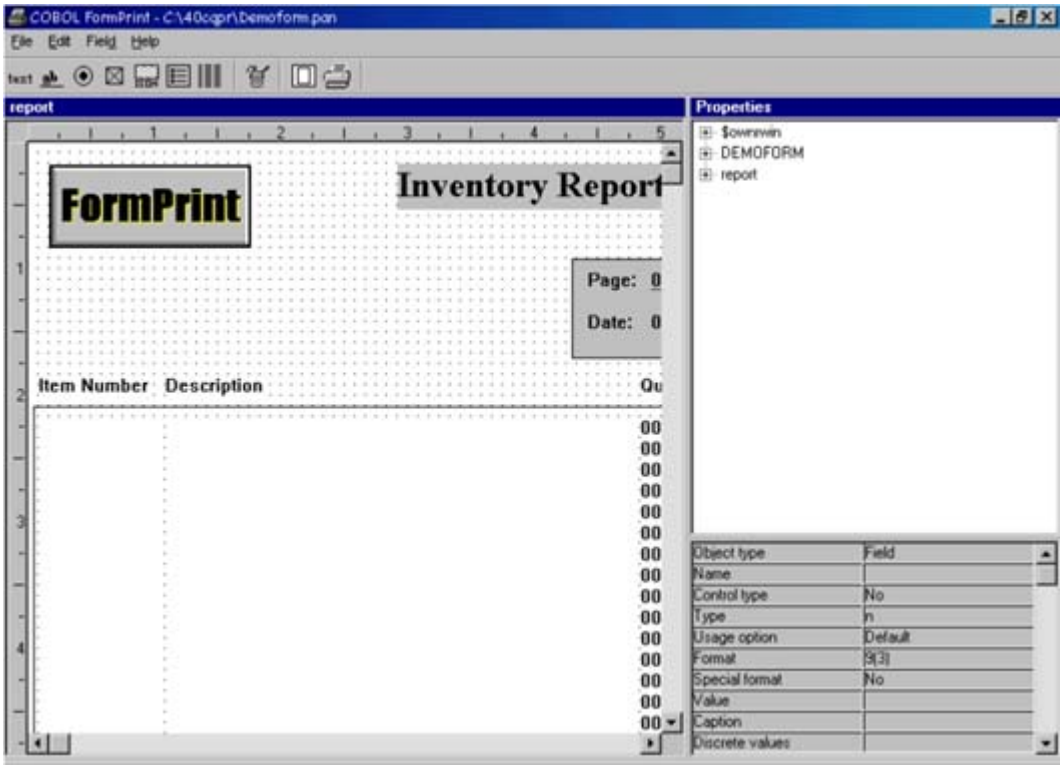
Click on the Border type Property and a drop down list will display the various border type choices. Select the “No = no border" option in the drop down list to eliminate the border. You may also use the (ARROW) keys in the Properties Box drop down list to select the appropriate option.

Press the (ENTER) key and then the (ESC) key to return to the Format Window or you can move the mouse pointer back to the Format Window and click once more. The border should disappear.

Repeat this process for the Report Date Field as well as the Item Number, Description, Quantity and Price Fields in the Repeat Field Group.

If you prefer to remove the border for the Field Repeat Group, you may do so as well, but for purposes of this tutorial, it will be left as the default border.

The Inventory Report Form should look like this now:



PLEASE NOTE: You may notice that there are three other border types, including default border, old 3D Border and Thin 3D Border. The FormPrint form editor allows you to replace the default border with a 3-dimensional border or thin 3-dimensional border for Custom Data Fields, Field Groups and Repeat Fields.

CHAPTER B12
Working with Check Boxes

What is a Check Box?

☒ Option

A Check Box is an option field which prints a symbol and associated text. The Check Box symbol has two different properties, “checked” and “unchecked” which indicates whether or not the associated text option has been selected. Check Boxes are generally used when you must provide the recipient of the form with information on which options have been selected from a limited set of choices. It is important to note that Check Boxes are generally used when any number of options may be selected as opposed to Radio Buttons which are generally mutually exclusive choices.

By moving the correct value to the program data area which represents the Check Box, you can then decide if a “checked” or an “unchecked” check box should be printed.

Add a Check Box to the Form

To create a Check Box field, click the mouse on the "Check Box" icon and click just below the left corner of the Inventory Report Static Text Field.

To establish a Check Box field using the keyboard, first position the cursor beneath the left corner of the Inventory Report Static Text Field and select the Field/Checkbox Menu Bar Option.

Assign a Caption to the Check Box

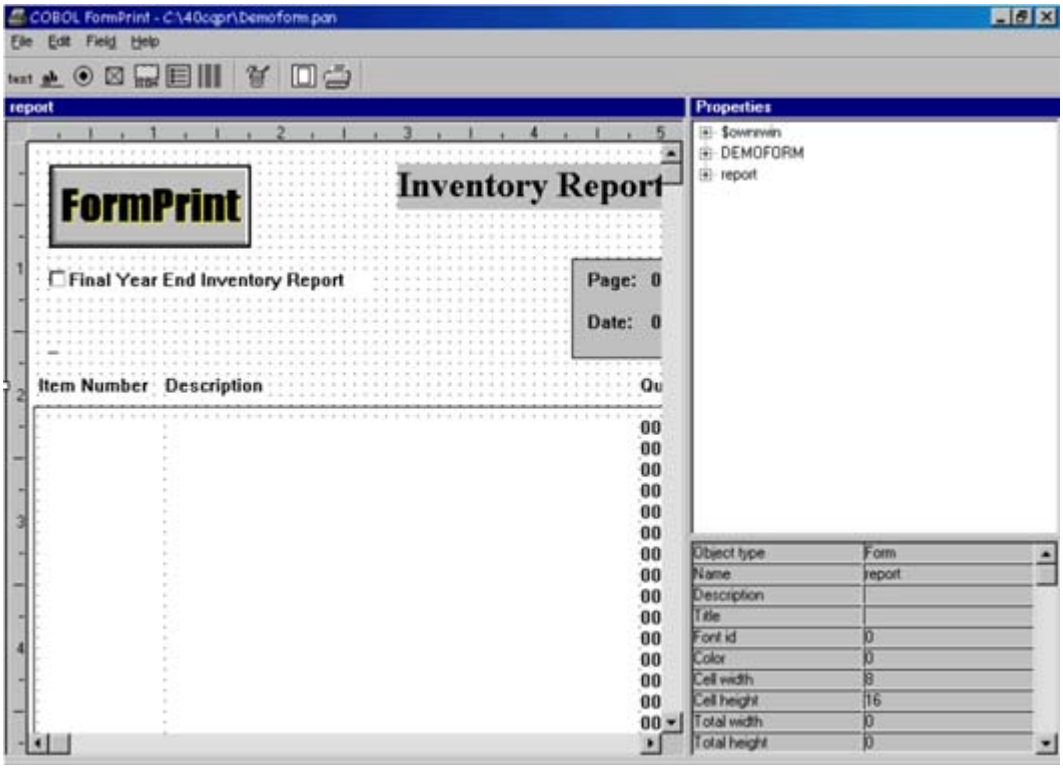
Select the Caption Property in the Properties Box. Change the value in the Caption Property from the default value of "Option" to "Final Year End Inventory Report." Press the (ENTER) key and then the (ESC) key to return to the Format Window.

Assign Program Values to the Check Box

Position the cursor in the Discrete Values Property in the Properties Box. A drop down list of entry fields will be displayed. Type "Y", then press the (DOWN ARROW) key to position the cursor in the next entry field, then type "N." Press the (ENTER) key and then the (ESC) key to return to the Format Window.

To turn the Check Box on from your program, set the generated Check Box field to "Y", to turn it off, set the field to "N".

The Inventory Report Form should now look like this:



CHAPTER B13
Working with Radio Buttons

What is a Radio Button?

◉ Yes

A Radio Button is an option field which prints a symbol and associated text. The Radio Button symbol has two different properties, “on” and “off” which indicates whether or not the associated text option has been selected. Radio Buttons are typically used when you must provide the recipient of the form with information on which single option has been selected from a limited set of choices. It is important to note that Radio Buttons are generally used when only one option may be selected as opposed to Check Boxes which are normally allow one or more options to be selected.

By moving the correct value to the program data area you can then decide if an “on” or an “off” Radio Button should be printed on your form.

Add a Radio Button to the Form

To create a Radio Button field, click the mouse on the "Radio Button" icon, move the pointer to the Format Window and click once just below the Check Box Field you created in the last chapter.

If you prefer to use the keyboard to create a Radio Button Field, first position the cursor. Select the Field/Radiobutton Menu Bar Option to create the default Radio Button Field.

Assign a Caption to the Radio Button

Select the Caption Property in the Properties Box. Change the value of the Caption Property from the default value of “Yes” to the value of "Pennsylvania Warehouse."

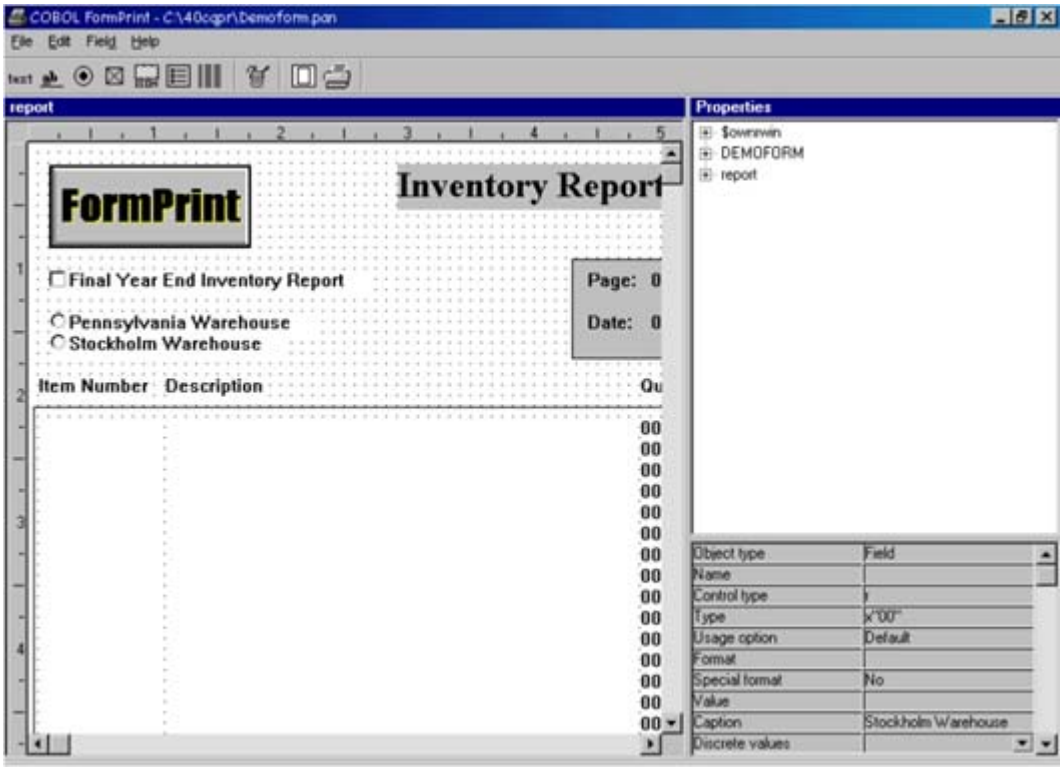
Create a second Radio Button directly below the first and assign it a Caption Property of “Stockholm Warehouse.”

Assign Program Values to the Radio Button

Position the cursor in the Discrete Values Property in the Properties Box. A drop down list of entry fields will be displayed. Type "Y", then press the (DOWN ARROW) key to position the cursor in the next entry field, then type "N." Press the (ENTER) key and then the (ESC) key to return to the Format Window.

To turn the Radio Button on from your program, set the generated Radio Button field to "Y", to turn it off, set the field to "N".

The Inventory Report Form should now look like this:



CHAPTER B14
Guidelines for Creating Forms

Introduction

The Form Editor makes it a fairly simple task to design forms. As you become more familiar with the editor, however, you may be tempted to create forms that are more and more sophisticated. The following information will make your task a little easier.

Spacing of Items within the Form

As you use the Forms Editor, you will notice that items which you place in the form tend to snap to the grid which fills the Form Editor window. This makes it easy to line items up within the form but may make it hard to achieve the desired spacing between items. To avoid this problem, you can reset the cell size properties (Cell width and Cell height Properties) to a lower value than the default size of 8 pixels by 16 pixels. If you set the size to 1 by 1, you will be able to position at any point within the form. It is best, however, to use as big a cell size as possible because the smaller the cell size, the more tedious it is to align items, unless you use the Row and Column Properties to position fields.

Lines and Boxes

Individual lines can be created using static fields, setting the type of the field to be a line rather than text. However, lines on a form are usually in the form of boxes around different areas on the form. It is normally best to use a group to create such a box. If you have multiple adjacent boxes, as in a spreadsheet for instance, use multiple groups. It does not matter if the edge of one group overlays the edge of the next group.

Fields are by default created with boxes (borders) around them. Normally it is appropriate to delete these borders. If you are creating a form that uses individually boxed fields, again like a spreadsheet, consider retaining these field borders. You can often use a repeat group to generate multiple instances of a bordered field to create the spreadsheet look.

Spreadsheets and Grids

When defining a large grid, the first method above (multiple group boxes) may run into trouble - FormPrint cannot deal with so many individual fields and groups. The second method (repeat group with bordered fields) works better but it is sometimes hard to get the field borders overlapping each other so that the grid looks nice and clean.

There is an option for repeat groups that makes this task a little easier. If the vertical or horizontal gap properties are set to -1 and the form Cell width and Cell height properties are set to 1 by 1, the appropriate field borders will overlap properly. Once the repeat group has been defined in this way, resizing the first field will resize the rest of the grid cells automatically. If you have multiple base fields in the grid, be sure to give all these fields the same height.

Colors and Shading

Individual items can be colored and the color will be reflected on the printed form either by shading or the actual color if you are using a color printer. The most common use of color, however, is to provide shading for a group box. To do this, set the color of the group and the fields within the group.

Foreground and background colors may be either solid or RGB values in FormPrint. In addition, the background color of a field may be set to a value of “transparent” so that fields may be overlaid on top of bitmap fields to allow the bitmap image to represent the background of the form.

Fonts

To ensure that the printed form matches as close as possible the form as defined using the Form Editor, it is best to only use True Type fonts. The issue of font consistency across printers and operating systems is discussed in Chapter B15.

Repeating Fields

If you have a lot of data fields on a form, consider using repeat groups wherever possible. It is much easier to define one field and repeat it than to define the same field multiple times.

Data Field Formatting

Field formatting is defined using COBOL picture clauses and is fairly comprehensive. This includes numeric formatting (Z,ZZ9.99, etc.), date formatting (MM/DD/YY, etc.) and special formats (999-99-9999, etc.). Use this capability wherever possible rather than trying to reformat data within your program.

Radio Buttons and Check Boxes

FormPrint supports the printing of check boxes and radio buttons on your forms. It is possible to use the same logic to check whether a radio button or check box is on or off. Radio button fields use two bitmaps which must be distributed with your final forms based application. The names of these buttons are RON.BMP (radio button on) and ROFF.BMP (radio button off). These should be placed in the current directory or a directory that is defined in the configuration variable QPRDIR. See Appendix A for details on configuration variables.

Bitmaps

You can use bitmaps in your forms. The format is DIB (device independent bitmap) and bitmap files should have the extension .BMP. These files should be placed in the current directory or a directory that is defined in the configuration variable QPRDIR. See Appendix A for details on configuration variables.

JPG format images may also be used, providing that you obtain the alternate bitmap add-on from Flexus. Contact your distributor if you prefer to use JPG images for your printed forms.

CHAPTER B15
Font Selection and Consistency

Introduction

Form Print allows you to create sophisticated forms featuring a variety of fonts and graphics and render those forms on any printer supported by Windows. Rendering graphics such as lines and pictures on a printer is usually fairly straightforward because these items are predefined collections of pixels which just need to be scaled as appropriate to fit the resolution of the printer. Fonts, unfortunately, are more complex because they rely on drivers to interpret their definitions and render them as appropriate. The end result is that a font defined in a certain way may be rendered differently both on the screens of different operating systems and on different printers regardless of the operating system. Form Print uses various techniques to deal with these potential differences and this chapter discusses those techniques and how to apply them.

Default Font Selection Process

After a font has been selected from the Font Dialog Box, Form Print records its typeface, size and style. These details are used to select the font when it is applied to a field or text in the form being designed and also to select an appropriate font for printing. When a new printer font is needed, a screen font is selected first and the dimensions of that font rechecked so that the printer font will be consistent with the screen font. The dimensions are then scaled up to the resolution of the printer and a printer font selected. The printer font is checked for consistency by taking a string of characters and checking that the total width of this string rendered using the screen font is consistent with the width of the string as rendered on the printer. If not, the width of the printer font is adjusted as appropriate.

Finer Control over Font Size

One problem with basing printer fonts on corresponding screen font definitions is that the size of the printer font becomes subject to the resolution of the screen. This is an issue because the resolution of most printers is higher than most screen resolutions and it is not possible to take advantage of this higher resolution to achieve fine control of font size. For example, suppose you need to print at 10 characters per inch and your printer's resolution is 600 dots (pixels) per inch. This means that you need a printer font width of 60 pixels, which is fine. However, the resolution of the screen is only 96 pixels per inch so you need a screen font width of 9.60 pixels to achieve 10 characters per inch, which is not possible because size has to be expressed in a whole number of pixels. The solution to this problem is a special font property called Font width increment which allows you to specify a pixel fraction to be added to the width for purposes of calculating the printer font width but can be ignored when selecting the screen font. In the editor, specify this increment using the Font change dialog box, accessible from the font list pulldown. When using Font width increment, it is important to also set the QPRFON x3 switch (see Appendix A) which allows an extra pixel or pixels to be inserted between each character in case the printer being used doesn't support the exact font width required.

Consistency between Printers

As explained above, the default font selection process can cause fonts to be rendered inconsistently across printers for two reasons: the screen fonts on which the printer fonts are based are rendered inconsistently across operating system versions; and the different printer drivers render fonts inconsistently on the printers that they support. In order to avoid the problems with inconsistent screen fonts, Form Print allows you to specify that a printer font be selected purely on the basis of its dimensions without regard to the size of a corresponding screen font. To do this, set the QPRFON x7 switch (see Appendix A). This configuration switch specifies that a font should be selected by taking a string of characters and checking that the length of the string rendered in the selected font (length A) is equal to the number of characters multiplied by the font width (length B). This technique tends to produce much more consistent fonts than the default process because length B is a number that does not vary across environments. It works especially well with fixed pitch fonts and quite well with variable pitch fonts. One problem with variable fonts is that the size of the resultant font can be very dependent on the characters that make up the test string The default characters are "abcdeghknopqsuxyz" and this works fairly well for most fonts. There are situations, however, where a different string might work better. For example, if most of your text is in upper case, it might be better to use upper case letters in the string. To use an alternative string, set the configuration variable QPRSTR (see Appendix A) to the value of the new string. The QPFON x3 switch does not have to be set if the x7 switch is set - x3 switch processing is assumed.

Consistency between Screen and Printer

Using the QPRFON x7 switch solves a lot of the problems with printer fonts by bypassing screen fonts entirely. Unfortunately, this introduces another problem which is that text that is viewed initially using the Preview facility (see Chapter C6) and thus must be rendered using screen fonts screen fonts may look quite different from the way it looks when it's eventually printed. This is mainly because, as mentioned above, the resolution of the screen is usually much less than that of the printer. To go some way to solving this problem, set the QPRFON x8 switch (see Appendix A) which causes additional pixels to be inserted at various character intervals to achieve better matching with printed output. This contrasts with the processing done for the QPRFON x3 switch (see Finer Control over Font Size above) which inserts pixels between every character. The x7 switch must be set for the x8 switch to take effect.

Finer Control over Vertical Position

Vertical position of text is not really a font issue because vertical position is controlled in Form Print by Cell size and Row. However, it will be discussed here as something which, like fonts, effects placement of text. As with fonts, positioning printer output or controlling lines per page is dependent on the respective resolutions of the screen and the printer. For example, to display 10 lines per inch on the screen, you would need a row height of 96 pixels divided by 10 which is not an exact number. Yet it is not a problem to print 10 lines per inch using a printer with a resolution of 600 dots per inch - each row needs to be 60 dots high. The solution to this problem is a special form property called Cell height increment which allows effective cell height to be expanded on the printer but left as a whole number of pixels on the screen. In this example, Cell height would be set to 9 and Cell height increment would be set to 60, allowing exactly 10 lines per inch for printed output.

PART C

Programming with COBOL FormPrint

CHAPTER C1
Introduction to Programming

This part of the user guide explains how to program with COBOL FormPrint. Because COBOL FormPrint makes printing forms inherently simple, once you have defined your form, as explained in Part B, there is generally not much left to do. Chapter C1 introduces programming with COBOL FormPrint.

Chapters C2 and C3 discuss code that can be generated by the forms editor. Subsequent chapters discuss other miscellaneous issues.

Calling COBOL FormPrint

All CALLs to the COBOL FormPrint runtime are made in the following format:

`CALL "QPR" USING function-area parameter-area.`

Function Area

The function specifies the type of operation you wish the runtime to perform. For example:

`SELECT-PRINTER` or **`PRINT-PAGE`**

The function parameter holds information which the runtime references during the operation. For example, the SELECT-PRINTER function parameter allows you to select the specific printer device to be used in the print job.

The function areas and parameter areas that are used are stored in a copy book called QPR.CPY.

The function area looks like this:

```
01  QPR-FUNCTIONS.
    05  QPR-SELECT-PRINTER    PIC S9(4) COMP-5 VALUE 0.
    05  QPR-PRINT-PAGE        PIC S9(4) COMP-5 VALUE 1.
    05  QPR-END-PRINT         PIC S9(4) COMP-5 VALUE 2.
    05  QPR-ABOUT             PIC S9(4) COMP-5 VALUE 3.
    05  QPR-INIT              PIC S9(4) COMP-5 VALUE 4.
    05  QPR-SELECT-PRINTER-EX PIC S9(4) COMP-5 VALUE 6.
```

How to Initialize FormPrint and Open a Form File

The first thing to be done is to Initialize FormPrint and open a forms file. This is done with the function QPR-INIT and the parameter QPR-INIT-AREA.

```
MOVE LOW-VALUES TO QPR-INIT-AREA.
MOVE "n" TO QPR-METHOD.
MOVE "AFORM.PAN" TO QPR-FILE.
CALL "QPR" USING QPR-INIT
                 QPR-INIT-AREA.
```

It's very important to initialize the areas with LOW-VALUE before moving values to them.
QPR-METHOD should always be set to 'n'.
QPR-FILE should contain the name of the forms file to be opened.

How to Select a Printer

Before actual printing can begin, a printer must be selected. The easiest way to do this is to select the default printer:

```
MOVE LOW-VALUES TO QPR-AREA.
```

```
MOVE "d" TO  QPR-DIALOG.
CALL "QPR" USING QPR-SELECT-PRINTER
                 QPR-AREA.
```

Initialize QPR-AREA to LOW-VALUES.

Set QPR-DIALOG to 'd' to select the default printer.

See chapter C4 for details on selecting the printer in other ways.

How to Print a Form

Once the printer has been selected, a form can be printed:

```
MOVE LOW-VALUES TO form-DATA
MOVE LOW-VALUES TO form-FIELDS
MOVE LOW-VALUES TO form-COLRS
MOVE "form" TO  form-NEXT-PANEL
```

Initialize the generated form data area and set the form name. At this point, you can set the values of any of the items in form-FIELDS. These data items correspond to the values of the fields within the form.

```
CALL "QPR" USING QPR-PRINT-PAGE
                 form-CONVERSE-DATA.
```

Print the form. The same form can be printed multiple times or other forms can be printed without the need to select the printer again.

How to Finish Up

When all forms have been printed, the printer must be deselected and the forms file closed:

```
CALL "QPR" USING QPR-END-PRINT
                 QPR-NULL-PARM.
```

If you need to print more forms after making the call, you must reinitialize FormPrint and reselect the printer before continuing.

CHAPTER C2
The Generated Program Data Division

This chapter describes items found in the Working-Storage section of the generated program.

Function Codes and Parameter area

```
COPY "QPR.CPY".
```

This COBOL copy-file contains all function codes and parameter areas needed to use FormPrint. The expanded copy looks like this:

```
*****************************
* Standard copy for QPR.DLL *
*****************************

01  QPR-FUNCTIONS.
    03  QPR-SELECT-PRINTER         PIC S9(4) COMP-5 VALUE 0.
    03  QPR-PRINT-PAGE             PIC S9(4) COMP-5 VALUE 1.
    03  QPR-END-PRINT              PIC S9(4) COMP-5 VALUE 2.
    03  QPR-ABOUT                  PIC S9(4) COMP-5 VALUE 3.
    03  QPR-INIT                   PIC S9(4) COMP-5 VALUE 4.
    03  QPR-SELECT-PRINTER-EX      PIC S9(4) COMP-5 VALUE 6.

01  QPR-AREA.
    05  QPR-RET-CODE               PIC S9(4) COMP-5.
    05  QPR-DATA.
        10  QPR-DIALOG             PIC X.
        10  QPR-DOC-NAME           PIC X(30).
        10  QPR-ORIENTATION        PIC X.
        10  QPR-COPIES             PIC S9(4) COMP-5.
        10  QPR-STRETCH            PIC X.
        10  QPR-DEVICE-LIST        PIC X(800).
        10  FILLER REDEFINES QPR-DEVICE-LIST.
            15  QPR-DEVICE-TO-SELECT
                                   PIC X(256).
            15  QPR-PRINT-FILE-NAME
                                   PIC X(256).
        10  QPR-DRIVER             PIC X(32).
        10  QPR-OUTPUT             PIC X(32).
        10  QPR-DATA-EX.
            15  QPR-PAPER-SIZE     PIC X.
            15  QPR-PAPER-SOURCE   PIC X.
            15  QPR-PREVIEW        PIC X.
            15  QPR-LEFT-MARGIN    PIC S9(4) COMP-5.
            15  QPR-TOP-MARGIN     PIC S9(4) COMP-5.
            15  QPR-ALLOW-COPIES   PIC X.
            15  QPR-DUPLEX         PIC X.
            15  FILLER             PIC X(8).
            15  QPR-PAPER-LENGTH   PIC S9(4) COMP-5.
            15  QPR-PAPER-WIDTH    PIC S9(4) COMP-5.
            15  QPR-PRINT-TO-FILE  PIC X.
            15  FILLER             PIC X(80).
01  QPR-VERSION.
    05  QPR-QE-RET                 PIC S9(4) COMP-5.
    05  QPR-QE-DATA.
        10  QPR-VERSION-INFO       PIC X(80).
        10  QPR-DISPLAY-BOX        PIC X.

01  QPR-INIT-AREA.
    05  QPR-QI-RET                 PIC S9(4) COMP-5.
    05  QPR-QI-DATA.
```

```
        10  QPR-LIBRARY            PIC X.
        10  QPR-METHOD             PIC X.
        10  QPR-FILE               PIC X(80).
        10  FILLER                 PIC X(118).

01  QPR-NULL-PARM.
    03 QPR-NP-RET-CODE             PIC S9(4) COMP-5.
```

Parameter for PRINT-PAGE Function

```
COPY "form.CPY".

*******************************
* parameter for PRINT-PAGE     *
*******************************
 01  form-CONVERSE-DATA.
     05  form-RET-CODE
                                   PIC S9(4) COMP-5.
     05  form-LENS.
         10  form-LEN-LEN
                                   PIC S9(4) COMP-5 VALUE +20.
         10  form-IP-NUM-LEN
                                   PIC S9(4) COMP-5 VALUE +40.
         10  form-IP-CHAR-LEN
                                   PIC S9(4) COMP-5 VALUE +104.
         10  form-OP-NUM-LEN
                                   PIC S9(4) COMP-5 VALUE +6.
         10  form-OP-CHAR-LEN
                                   PIC S9(4) COMP-5 VALUE +2.
         10  form-FIELD-LEN
                                   PIC S9(4) COMP-5 VALUE +2121.
         10  form-COLR-LEN
                                   PIC S9(4) COMP-5 VALUE +178.
         10  FILLER
                                   PIC S9(4) COMP-5 VALUE +0.
         10  FILLER
                                   PIC S9(4) COMP-5 VALUE +0.
         10  FILLER
                                   PIC S9(4) COMP-5 VALUE +0.
     05  form-DATA.
******** form-IP-NUM-DATA ********
         10  form-KEY
                                   PIC S9(4) COMP-5.
         10  FILLER                PIC X(38).
******** form-IP-CHAR-DATA *******
         10  form-NEXT-PANEL
                                   PIC X(8).
         10  FILLER                PIC X(94).
         10  form-WAIT-SW
                                   PIC X.
         10  FILLER                PIC X.
******** form-OP-NUM-DATA ********
         10  form-PAN-POS-SW
                                   PIC S9(4) COMP-5.
         10  form-PAN-ROW
                                   PIC S9(4) COMP-5.
         10  form-PAN-COL
                                   PIC S9(4) COMP-5.
******** form-OP-CHAR-DATA ********
         10  FILLER                PIC X(2).
******** form-OP-VAR-DATA ********
```

```
     05  form-FIELDS.
         10  form-PAGE-NUMBER
                                    PIC 9(003).
         10  form-REPORT-DATE
                                    PIC 9(0008).
         10  form-FIELD-ID-104
                                    PIC X(0001).
             88  form-FIELD-ID-104-YES
                                    VALUE "0".
             88  form-FIELD-ID-104-NO
                                    VALUE "1".
         10  form-FIELD-ID-105
                                    PIC X(0001).
             88  form-FIELD-ID-105-YES
                                    VALUE "0".
         10  form-FIELD-ID-106
                                    PIC X(0001).
             88  form-FIELD-ID-106-YES
                                    VALUE "0".
         10  FILLER OCCURS 43.
             15  form-ITEM-NUMBER
                                    PIC X(0012).
             15  form-DESCRIPTION
                                    PIC X(0025).
             15  form-QUANTITY
                                    PIC 9(005).
             15  form-PRICE
                                    PIC 9(05)V9(02).
     05  form-COLRS.
         10  form-PAGE-NUMBER-C
                                    PIC X.
         10  form-formprnt-C
                                    PIC X.
         10  form-REPORT-DATE-C
                                    PIC X.
         10  form-FIELD-ID-104-C
                                    PIC X.
         10  form-FIELD-ID-105-C
                                    PIC X.
         10  form-FIELD-ID-106-C
                                    PIC X.
         10  FILLER OCCURS 43.
             15  form-ITEM-NUMBER-C
                                    PIC X.
             15  form-DESCRIPTION-C
                                    PIC X.
             15  form-QUANTITY-C
                                    PIC X.
             15  form-PRICE-C
                                    PIC X.
*************
* field ids *
*************
 01  form-IDS.
         05  form-PAGE-NUMBER-I
                                    PIC S9(4) COMP-5 VALUE +2.
         05  form-formprnt-I
                                    PIC S9(4) COMP-5 VALUE +1.
         05  form-REPORT-DATE-I
                                    PIC S9(4) COMP-5 VALUE +3.
         05  form-FIELD-ID-104-I
                                    PIC S9(4) COMP-5 VALUE +104.
         05  form-FIELD-ID-105-I
```

```
                                PIC S9(4) COMP-5 VALUE +105.
    05  form-FIELD-ID-106-I
                                PIC S9(4) COMP-5 VALUE +106.
    05  form-ITEM-NUMBER-I
                                PIC S9(4) COMP-5 VALUE +4.
    05  form-DESCRIPTION-I
                                PIC S9(4) COMP-5 VALUE +5.
    05  form-QUANTITY-I
                                PIC S9(4) COMP-5 VALUE +6.
    05  form-PRICE-I
                                PIC S9(4) COMP-5 VALUE +7.
```

This parameter is used when calling FormPrint with function QPR-PRINT-PAGE. The forms name precedes all the data items so that multiple forms can be processed in the sample program. In this program one data item is referenced: NEXT-PANEL is set to the name of the form to be printed.

You are not permitted to alter any of the length items in the form-LENS area.

If you wish to suppress the generation of the forms name prefix, edit the UIB.CBX file to remove all "$001-" strings from the file, except on those lines containing an 01-level.

CHAPTER C3
The Generated Program Procedure Division

This chapter describes the logic found in the Procedure Division of the generated program. This logic is quite simple but illustrates the code that must be included in a program to print a form.

Initializing FormPrint

```
     PERFORM INIT.

**************
 INIT SECTION.
**************

     MOVE LOW-VALUES TO QPR-INIT-AREA.

     MOVE "n"         TO QPR-METHOD.
     MOVE "AFORM.PAN"       TO QPR-FILE.

     CALL "QPR" USING
                QPR-INIT
                QPR-INIT-AREA.

     IF QPR-QI-RET NOT = 0
         STOP RUN
     END-IF.
```

It's very important to initialize the areas with LOW-VALUE before moving values to them.

QPR-METHOD must be set to 'n'.

QPR-FILE should contain the name of the forms file to be opened.

Selecting the printer

```
     PERFORM SELECT-PRINTER.

************************
 SELECT-PRINTER SECTION.
************************
     MOVE    LOW-VALUES    TO  QPR-AREA.

     MOVE "Document 1"     TO  QPR-DOC-NAME.
     MOVE "p"              TO  QPR-ORIENTATION.
     MOVE "y"              TO  QPR-DIALOG.
```

Initialize QPR-AREA to LOW-VALUES. Move the name of the document to QPR-DOC-NAME. Decide what orientation to use and choose a method of selecting the printer. In this example, we have set QPR-ORIENTATION to 'p' for portrait and set QPR-DIALOG to 'y' to invoke the Windows Print Dialog box. We then issue the call to select the printer:

```
     CALL "QPR"   USING     QPR-SELECT-PRINTER
                            QPR-AREA.
```

Printing forms

```
     PERFORM PRINT-PAGES.

********************
 PRINT-PAGES SECTION.
```

```
*********************
     MOVE    LOW-VALUES      TO  $001-DATA
     MOVE    LOW-VALUES      TO  $001-FIELDS
     MOVE    LOW-VALUES      TO  $001-COLRS
     MOVE    "form"          TO  $001-NEXT-PANEL.

* add logic here to fill variables declared in form-fields

     CALL    "QPR"           USING
                             QPR-PRINT-PAGE
                             form-CONVERSE-DATA.
```

Before you issue the QPR-PRINT-PAGE call, you can set the values of the fields that are going to be printed. This is done by moving values to the items in the fields area, form-FIELDS.

Ending the print session

```
     PERFORM END-PRINT.

*******************
 END-PRINT SECTION.
*******************
    CALL "QPR"  USING  QPR-END-PRINT
                       QPR-NULL-PARM
```

CHAPTER C4
Selecting the Printer with COBOL FormPrint

Methods for Selecting the Printer

Chapter 1 described how to select the default printer and Chapter 4 described using the Print Dialog Box to select the printer. This chapter discusses these two methods in a little more detail and describes still one more method.

The following are the three methods for selecting the printer:

Method 1 - Use the Windows Print Dialog Box and allow the user to select the printer
Method 2 - Select the default printer
Method 3 - Get a list of available printers and select a specific printer

Method 1 - Using the Windows Print Dialog Box

```
1. MOVE LOW-VALUES TO QPR-AREA.
   MOVE "y" TO QPR-DIALOG.
```

Move "y" to the QPR-DIALOG parameter to display the Print Dialog Box.

```
2. MOVE "< document name >" TO QPR-DOC-NAME.
```

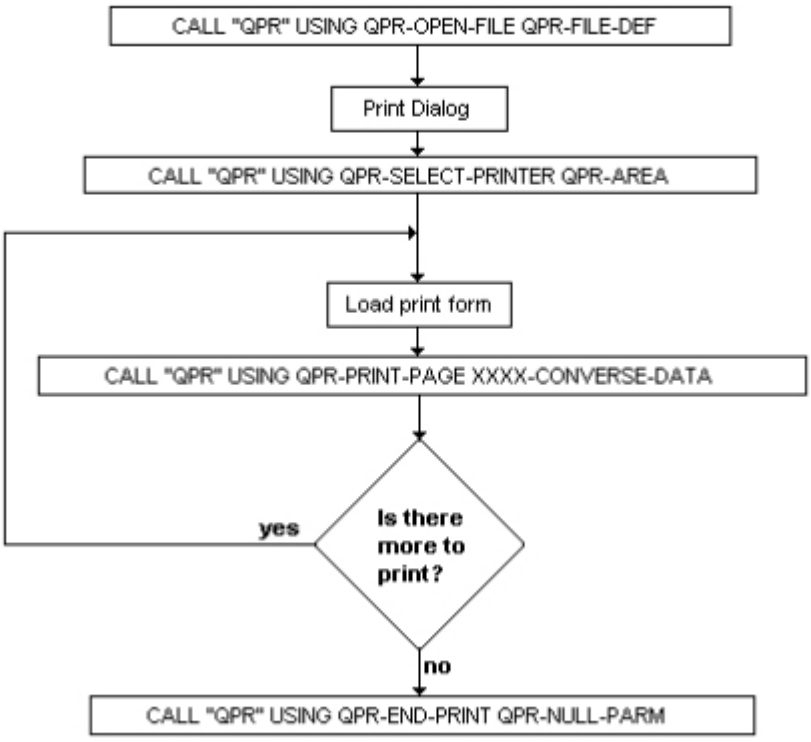
Enter the name of your document, if you want it to be displayed with the name in the print queue.

```
3. CALL "QPR" USING QPR-SELECT-PRINTER QPR-AREA.
```

CALL QPR using the QPR-SELECT-PRINTER function parameter and the QPR-AREA data parameter.

Using this method, the user can set paper orientation, paper size and other options supported by the default printer driver.

Method 1 – Flowchart



Method 2 - Selecting the Default Printer

1. `MOVE LOW-VALUES TO QPR-AREA.`
   `MOVE "d" TO QPR-DIALOG.`

   Move "d" to QPR-DIALOG to choose the default printer. Setting QPR-DIALOG TO LOW-VALUE will also cause the default printer to be chosen.

2. `MOVE "<document name>" TO QPR-DOC-NAME`

   Enter the name of your document if you want the document name to be displayed in the print queue.

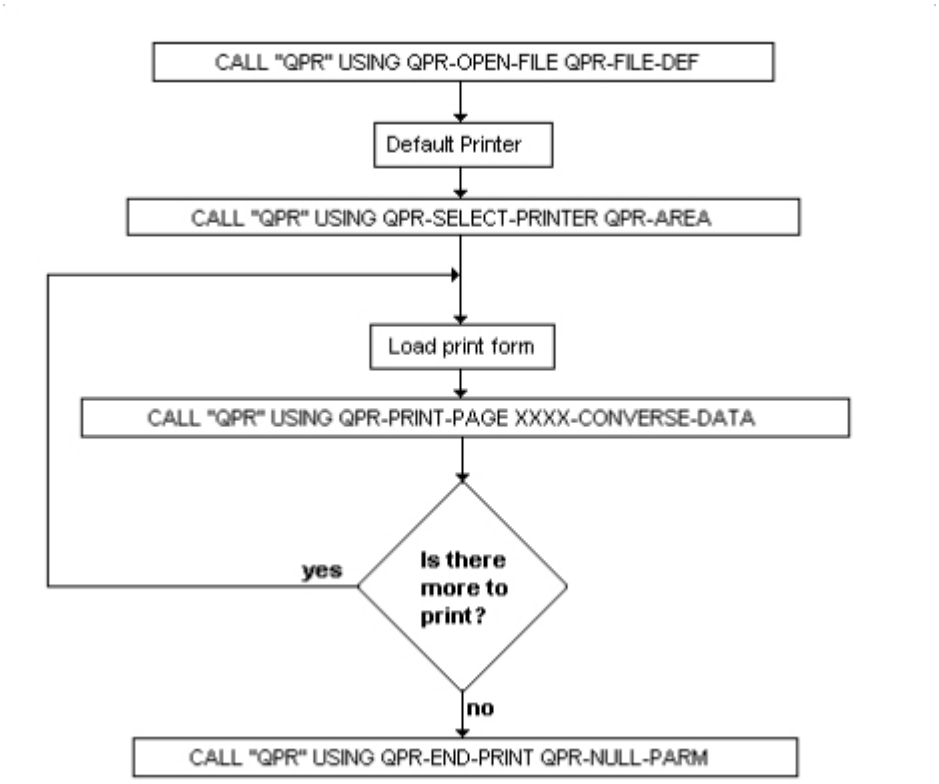
3. `MOVE  "l" TO QPR-ORIENTATION.`

. Choose paper orientation by moving "p" (Portrait), "l" (Landscape) to the field QPR-ORIENTATION. Portrait is default.

4. `CALL "QPR" USING QPR-SELECT-PRINTER QPR-AREA.`

   Call FormPrint with the function QPR-SELECT-PRINTER and the parameter QPR-AREA.

Method 2 - Flowchart



Method 3 – Selecting a Specific Printer

```
1. MOVE LOW-VALUES TO QPR-AREA.
   MOVE "s" TO QPR-DIALOG.
```

Move "s" to QPR-DIALOG to select a printer. If you know a specific printer is available, you can bypass steps 2 and 3 by setting QPR-RET-CODE to –10 and advancing to step 4.

```
2. CALL "QPR" USING QPR-SELECT-PRINTER QPR-AREA.
```

Call QPR to get the printer list. Return code is -10 if QPR managed to get printers. This must not be altered. QPR needs this value in the next call.

```
3. UNSTRING QPR-DEVICE-LIST DELIMITED BY ';' INTO
           WS1-DEVICES(WS1-I) WITH POINTER WS1-OFFSET.3.
```

FormPrint returns the list to QPR-DEVICE-LIST. Device names are separated with semicolon (;), and should be unpacked with UNSTRING.

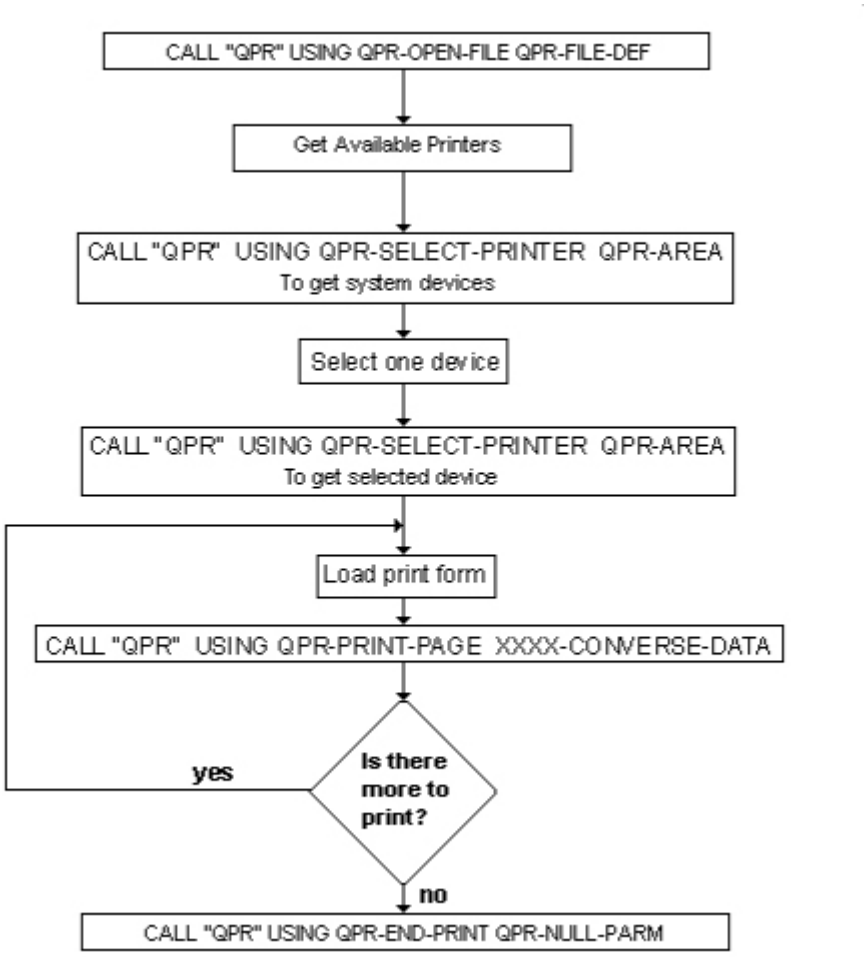
```
4. MOVE WS1-DEVICES (3) TO QPR-DEVICE-TO-SELECT.
```

Choose which printer you want to use:

```
5. CALL "QPR" USING QPR-SELECT-PRINTER QPR-AREA.
```

Select the printer.

Method 3 - Flowchart



Setting Additional Printer Properties

Additional printer properties can be set when selecting the printer by using the function QPR-SELECT-PRINTER-EX instead of QPR-SELECT-PRINTER. This function works in exactly the same way as QPR-SELECT-PRINTER except that it takes into account additional data items at the end of QPR-AREA. These are defined as follows:

```
01  QPR-AREA.
    05  QPR-RET-CODE          PIC S9(4)   COMP-5.
    05  QPR-DATA.
        ...
        10  QPR-DATA-EX.
            15  QPR-PAPER-SIZE PIC X.
                88  QPR-LETTER            VALUE X"01".
                88  QPR-LEGAL             VALUE X"05".
                88  QPR-A4                VALUE X"09".
                88  QPR-B5                VALUE X"0D".
            15  QPR-PAPER-SOURCE PIC X.
                88  QPR-UPPER-TRAY        VALUE X"01".
                88  QPR-LOWER-TRAY        VALUE X"02".
                88  QPR-MIDDLE-TRAY       VALUE X"03".
                88  QPR-MANUAL-FEED       VALUE X"04".
            15  QPR-PREVIEW   PIC X.
                88  QPR-DO-PREVIEW        VALUE "y".
```

```
15  QPR-LEFT-MARGIN
                    PIC S9(4) COMP-5.
15  QPR-TOP-MARGIN
                    PIC S9(4) COMP-5.
15  QPR-ALLOW-COPIES
                    PIC X.
15  QPR-DUPLEX     PIC X.
    88  QPR-DUP-SIMPLEX      VALUE X"01".
    88  QPR-DUP-VERTICAL     VALUE X"02".
    88  QPR-DUP-HORIZONTAL   VALUE X"03".
15  FILLER         PIC X(8).
15  QPR-PAPER-LENGTH
                    PIC S9(4) COMP-5.
15  QPR-PAPER-WIDTH
                    PIC S9(4) COMP-5.
15  QPR-PRINT-TO-FILE
                    PIC X.
15  FILLER         PIC X(80).
```

For example, you could use the following code to set the paper size:

```
MOVE LOW-VALUES TO QPR-AREA.
MOVE "d" TO QPR-DIALOG.
SET QPR-LEGAL TO TRUE.
CALL "QPR" USING QPR-SELECT-PRINTER-EX QPR-AREA.
```

For details on using Print Preview (QPR-PREVIEW = "y"), see Chapter C6. See chapter E3 for details on the other options.

CHAPTER C5
Dynamically Modifying the Form at Runtime

Prior to actually printing a form, it is normal to set the values of variable fields within the form by moving data to the items in the generated form-FIELDS area. This is done immediately before the PRINT-PAGE call and was explained in Chapter 1 and Chapter 3. Apart from setting the values of these variable fields, other properties of form objects can be easily set at runtime. These properties are as follows:

- Colors of fields
- Bitmap images
- Fonts

If you need to set other properties or even create form fields at runtime, please refer to Chapters C8 and C9.

Changing a Color

Suppose you wish to modify the color of the ACCOUNT-NUMBER field to highlight it in a report. Insert the following code before the PRINT-PAGE call but after the initialization of form-COLRS:

```
MOVE X"0B" TO form-ACCOUNT-NUMBER-C.
```

A –C item is generated for each of the fields in your form. The value moved to a –C item must be the hex representation of the id of color (in this case color id 11). Colors are established in the form editor.

Changing a Bitmap Image

Copy the qprbmpws.cpy file into your working-storage and the qprbmp.cpy file into your procedure division.

Before calling the QPR-PRINT-PAGE function, do the following for each field to be changed:

```
MOVE form-name TO QPRBMP-PANEL.
MOVE field-id TO QPRBMP-ID.
MOVE new-bitmap-name TO QPRBMP-NAME.
PERFORM SET-QPRBMP.
```

The field-id is a generated item at the end of the form-name.CPY file and is made up of the field-name with a -I suffix.

After QPR-PRINT-PAGE, do the following (once only):

```
PERFORM CLEANUP-QPRBMP.
```

Changing a Font

Copy the qprfntws.cpy and qprbmpws.cpy files into your working-storage and the qprfnt.cpy file into your procedure division.

Before calling the QPR-PRINT-PAGE function, do the following for each field to be changed:

```
MOVE form-name TO QPRBMP-PANEL.
MOVE field-id TO QPRBMP-ID.
MOVE new-font-id TO QPRFNT-FONT-ID.
PERFORM SET-QPRFNT.
```

The field-id is a generated item at the end of the form-name.CPY file and is made up of the field-name with a -I suffix.

The new-font-id is the number assigned to the desired font as displayed on the font listing in the editor. If the size of the new font is greater than the original font, remember to allow extra space when sizing the field in the editor.

After QPR-PRINT-PAGE, do the following (once only):

```
PERFORM CLEANUP-QPRFNT.
```

CHAPTER C6
Print Preview Facility

This facility allows users to view output from your program before it is actually printed.

Invoking the Print Preview Facility

Print Preview can be invoked either automatically by setting a configuration variable, or by setting an option in your program.

1. To invoke Print Preview automatically, set configuration variable QPRPVW to 1. See Appendix A for details of using configuration variables. All print output from your program will automatically be displayed for viewing by the user until the user exits the Print Preview facility. All print output after this will be sent directly to the printer until your program issues a QPR-END-PRINT call. If you are modifying forms at runtime (see Chapter C8) set QPRPVW=5 otherwise the changes will be lost during preview.

2. To invoke Print Preview programmatically, use QPR-SELECT-PRINTER-EX instead of QPR-SELECT-PRINTER and set QPR-DO-PREVIEW to TRUE (QPR-PREVIEW = "y") before making the call. As above, Print Preview will be active until the user exits the facility. If you are modifying forms at runtime (see Chapter C8) set QPR-PREVIEW = "d" otherwise the changes will be lost during preview.

Using the Print Preview Facility

Once Print Preview has been invoked, print output from your program will be displayed in a window rather than sent to the printer. This window has a menu and toolbar with the following options:

Get all pages - process all pages before previewing
Print - print selected pages and return to previewing
Cancel print - process all pages and discard
Exit - exit preview facility with the option to print pages so far
Find - search for a text string
Next page - preview next page
Previous page - preview previous page
First page - preview first page
Last page - preview last page processed so far

All print output after the exit option has been chosen will go directly to the printer. The exit option will automatically be invoked when a QPR-END-PRINT call is made by your program.

The Find facility will only search pages that have already been processed so use Get all pages first if you need to search all pages.

The Print menu option will cause either the system printer dialog box to be displayed (if QPR-DIALOG is "a") or a special Range dialog box (if QPR-DIALOG is not "a")

Configuring the Print Preview Facility

The QPRPVT configuration variable (see also Appendix A) allows menu options to be enabled and disabled and the text of menu options and messages to be set. QPRPVT should point to a text file that contains option and message ids and the corresponding text as follows:

001~File
021Get ~all pages
022E~xit preview
023~Cancel print
024~Print

```
002~View
041~Next page  ^PgDn#374
042~Previous page  ^PgUp#388
043~First page
044~Last page
003~Edit
061~Find…  F3#317
900Print preview
901Cancel print
902Are you sure?
903Exit preview
904Would you like to print previewed pages?
905Progress...
906Getting page
907Printing page
908String not found
909(Scroll up  PgUp)#329
910(Scroll down  PgDn)#337
911(Scroll left  ^Lt)#371
912(Scroll right  ^Rt)#372
913myrange.pan
```

The text may be modified as appropriate but the option numbers should be left intact.

To suppress an option, set the text to "N/A". If certain options are suppressed, this has special meaning, as follows:

021N/A - all pages will be automatically retrieved but the user will be allowed to interact with pages retrieved so far, while other pages are being retrieved.

900N/A - the text in the Preview title bar will be taken from the Title property of the form being previewed.

903N/A - selecting "Exit preview" from the menu will cause an immediate exit without a "Would you like to print?" message box.

If the text for 900 (window title) is preceded with a ">" or "^", the preview window will be maximized or non-owned respectively. The special characters will not be displayed.

Use "#" followed by a 3-digit numeric key code to specify key usage. The entries 909 through 912 are used solely for this purpose. For example:

```
041~Next page  PgDn#337
042~Previous page  PgUp#329
043~First page  ^PgUp#388
044~Last page  ^PgDn#374
909(Scroll up  Up)#328
910(Scroll down  Dn)#336
911(Scroll left  Lt)#331
912(Scroll right  Rt)#333
```

The text for entry 913 points to a file containing a definition of the Range dialog box. Use the Form Print editor to modify the text for this dialog box.

Programming for Print Preview

Apart from the code needed to invoke Print Preview, there is really nothing else that has to be done i.e. your code runs the same whether it is in Print Preview mode or regular print mode. However, there are some issues that you need to be aware of:

1. If you are dynamically modifying forms as explained in Chapter C5 or Chapter C8 or you are using multi-form pages as explained in Chapter C7 , you must set QPR-PREVIEW to "d" rather than "y" (or set configuration variable QPRPVW to 5 rather than 1).

2. If the user selects "Cancel print" before you have called the END-PRINT function, -17 will be returned from the PRINT-PAGE call in form-KEY (in form-CONVERSE-DATA). This allows you to suppress further PRINT-PAGE calls and issue an immediate END-PRINT call.

3. If the user selects "Exit" and then elects not to print previewed pages, -18 will be returned to indicate this. This value may be returned in one of two places: if the current call is PRINT-PAGE, the value will be returned in form-KEY; if the current call is END-PRINT, the value will be returned in QPR-NP-RET-CODE. This allows you to take special action if the hardcopy print is suppressed by the user. If you ignore the -18 return-code on a PRINT-PAGE call and make further PRINT-PAGE calls, print output resulting from these calls will be sent to the printer.

CHAPTER C7
Multi-Form Pages

Sometimes, not only the data printed on a page may vary but also the overall format of the page. This situation can usually be dealt with by using different forms for each page, but this solution is not always convenient. For example, suppose you have an invoice and the format of the detail lines varies depending on the item being listed. It may be impossible to define multiple forms to cover all possible combinations of detail lines on a single page. Although it makes programming more complicated, the only practical solution may be to use a separate form for each area within a page, in this case, a different form for each type of detail line.

Defining Forms for Multi-Form Pages

Forms to be used on multi-form pages are defined in the same way as any other form. Start the form definition at the top of the editor Format Window even though the form may eventually be printed further down the page.

Programming for Multi-Form Pages

Forms are printed as usual using the QPR-PRINT-PAGE function but with options set to tell QPR that a multi-form page is being output and to specify where on the page a form should be placed.

1. To specify that a multi-form page is being output, set form-WAIT-SW to "n". This will cause QPR to use the form to construct the page in memory but not output it to the printer. Before the call for the last form to be included on the page, reset form-WAIT-SW to low-value and the page will be printed.

2. To specify the position of the form on the page, set form-PAN-POS-SW to 1 and form-PAN-ROW and form-PAN-COL to the appropriate row and column (in cells). This should only be done for the second and subsequent forms - the first form is always positioned at the top of the page.

CHAPTER C8
Getting and Setting Properties

In Chapter C5 you were shown how to modify a form at runtime: changing colors, specifying images to be printed and changing fonts. These were in fact examples of resetting object properties at runtime. In most cases, it is possible to print your forms without ever accessing object properties directly. This is because very often the only thing you will need to do to a form at runtime is set the values of the variable fields in the form and this is done by merely moving data into the items in the generated form-fields area.

Sometimes, however, as with the example of resetting the font in Chapter C5, it is necessary to set an object property directly because there is no current form property available. In this case, the font property was being reset and a special copy file (QPRFNT.CPY) was used to do this. It is possible, however, to reset almost all of the properties of all the FormPrint objects (forms, fields, etc.) These properties are described in detail in part E.

This chapter describes the function calls needed to modify properties. The first part describes the traditional method which involves retrieving the definition of an object, resetting a particular property or properties, then updating the object definition. The second part describes the newer method which involves updating a particular property directly. This newer method should be used in new code when only a single property needs to be retrieved or modified.

Before properties can be accessed, your program must switch into " maintenance mode" using the following function call:

```
CALL "QPR" USING QPR-ENTER-MAINTENANCE-MODE QPR-NULL-PARM.
```

If you examine the contents of QPRFNT.CPY, you will see this call being made. The functions and parameters discussed in this chapter are defined in a copy file called QPRMAINT.CPY which must be included in your program.

Once in maintenance mode, remember that regular printing functions (PRINT-PAGE, etc.) cannot be used - to exit maintenance mode and return to printing mode, make the following call:

```
CALL "QPR" USING QPR-EXIT-MAINTENANCE-MODE QPR-NULL-PARM.
```

Before a form's properties can be modified, the form must be read into memory. Make the following call to do this:

```
MOVE LOW-VALUES TO QPR-WD-DATA.
MOVE "form" TO QPR-WD-PANEL-NAME.
MOVE "y" TO QPR-WD-HIDE-SW.
CALL "QPR" USING QPR-OPEN-WINDOW QPR-WINDOW-DEF.
```

You now have access to the form and all the objects within the form - fields, static fields, groups, repeat groups.

Accessing Properties Using Object Definitions

The technique used for resetting a property is to call a function to get the object and its properties, reset the data item related to the appropriate property, then call a function to reset the object. Here is a list of the objects and the functions used to access them:

Form -	GET/SET-PANEL-DEF
Field -	GET/SET-FIELD-DEF
Static field -	GET/SET-STATIC-DEF
Group -	GET/SET-GROUP-DEF
Repeat group -	GET/SET-REPEAT-DEF
Font -	GET/SET-FONT-DEF
Color -	GET/SET-COLOR-DEF

To reset the text in a static field, for example, you need to reset the Text property of the static field:

```
MOVE LOW-VALUES TO QPR-SD-DATA.
```

```
MOVE 5 TO QPR-SD-ROW.
MOVE 10 TO QPR-SD-COL.
CALL "QPR" USING QPR-GET-STATIC-DEF QPR-STATIC-DEF.
MOVE "New text" TO QPR-SD-TEXT.
CALL "QPR" USING QPR-SET-STATIC-DEF QPR-STATIC-DEF.
```

Notice that you must set the data item associated with the key of the object that is being accessed, prior to the GET call. Here is a list of the keys associated with each object:

Form -	PD-NAME
Field -	FD-ID or FD-NAME
Static field -	SD-ROW/COL or SD-ID
Group -	GD-ID
Repeat group -	RD-ID
Font -	FO-ID
Color -	CO-ID

Where alternate keys are listed, the alternate key may only be used if the primary key is set to low-values or zero. In the case of static fields, the alternate key (SD-ID) may only be used if the Id property for the static field is set when the static is created (in the editor it defaults to zero).

Accessing field properties is a special case because the format of the variable length field properties varies so much. The following code should be used:

```
MOVE 2000 TO QPR-FD-VAR-LEN.
MOVE LOW-VALUES TO QPR-FD-VAR-LENS.
MOVE LOW-VALUES TO QPR-FD-DATA.
MOVE ACCOUNT-NUMBER-I TO QPR-FD-ID.
CALL "QPR" USING QPR-GET-FIELD-DEF QPR-FIELD-DEF.
MOVE X"0C" TO QPR-FD-COLR.
CALL "QPR" USING QPR-SET-FIELD-DEF QPR-FIELD-DEF.
```

2000 is the length of FD-VAR-DATA as defined in QPR.CPY. This length may need to be increased if the combined length of the variable field properties exceed this length. Setting FD-VAR-LENS to low-values causes the variable-length property values to be returned as one concatenated string in FD-VAR-DATA rather than reformatted and split up as for form variable-length properties in PD-VAR-DATA. To access each property individually, you need to refer to the variable lengths returned in FD-VAR-LENS after a GET-FIELD-DEF call and unstring FD-VAR-DATA. Code to do this is provided in the following copy files:

FD-FMT -	QPRRSFMT.CPY
FD-CAPTION -	QPRRSCAP.CPY
FD-INITIAL-VAL -	QPRRSVAL.CPY

To reset the caption of a radio button, for example, you would use the following code:

```
COPY "QPRBMPWS.CPY"

01  QPRCAP-DATA PIC X(20).
MOVE 2000 TO QPR-FD-VAR-LEN.
MOVE LOW-VALUES TO QPR-FD-VAR-LENS.
MOVE LOW-VALUES TO QPR-FD-DATA.
MOVE RADIO-BUTTON-I TO QPR-FD-ID.
CALL "QPR" USING QPR-GET-FIELD-DEF QPR-FIELD-DEF.
MOVE "New caption" TO QPRCAP-DATA
MOVE 11 TO QPRBMP-LEN.
PERFORM QPRRSCAP.
CALL "QPR" USING QPR-SET-FIELD-DEF QPR-FIELD-DEF.

COPY "QPRRSCAP.CPY"
```

Some properties cannot be changed without deleting and recreating the object. This is the case with the Row and Column properties that control the location of an object. To move a field, for example, you must do the following:

```
MOVE 2000 TO QPR-FD-VAR-LEN.
MOVE LOW-VALUES TO QPR-FD-VAR-LENS.
MOVE LOW-VALUES TO QPR-FD-DATA.
MOVE ACCOUNT-NUMBER-I TO QPR-FD-ID.
CALL "QPR" USING QPR-GET-FIELD-DEF QPR-FIELD-DEF.
ADD 5 TO QPR-FD-ROW
ADD 10 TO QPR-FD-COL
CALL "QPR" USING QPR-DELETE-FIELD QPR-FIELD-DEF.
CALL "QPR" USING QPR-SET-FIELD-DEF QPR-FIELD-DEF.
```

Once your form's properties have been reset as appropriate, you must make the following call before printing the form:

```
CALL "QPR" USING QPR-EXIT-MAINTENANCE-MODE QPR-NULL-PARM.
```

Once your form has been printed, release the memory used to modify the form with the following code:

```
CALL "QPR" USING QPR-ENTER-MAINTENANCE-MODE QPR-NULL-PARM.
MOVE "form" TO QPR-ND-NAME.
CALL "QPR" USING QPR-ACTIVATE-WINDOW QPR-NAME-DEF.
CALL "QPR" USING QPR-CLOSE-WINDOW QPR-NULL-PARM.
CALL "QPR" USING QPR-EXIT-MAINTENANCE-MODE QPR-NULL-PARM.
```

Make sure you do this before making the END-PRINT call.

Accessing Properties Directly

The code at the start of this chapter to reset the Window Title could have been written like this:

```
MOVE LOW-VALUES TO QPR-PR-DATA.
MOVE 5 TO QPR-PR-ROW.
MOVE 10 TO QPR-PR-COL.
MOVE "SVA0000000008-RL" TO QPR-PR-KEY.
MOVE "New text" TO QPR-PR-VALUE.
CALL "QPR" USING QPR-SET-PROPERTY QPR-PROPERTY.
```

This code is a little more economical than the original code because it is no longer necessary to retrieve the property definition before resetting it (although of course there is a GET-PROPERTY function which corresponds to the SET-PROPERTY function). The code is fairly self-explanatory except for the fourth line. This is the code which tells the SET-PROPERTY function exactly which property is being accessed – the property Key. The rest of this chapter will discuss how to set this property Key. The only other thing you need to worry about is setting the Id of the object being accessed. For a static, row and column is used, as above. For a form, an Id is not needed. For a field, use the ids in the generated copy file, for example:

```
MOVE form-ACCOUNT-NUMBER-I TO QPR-PR-ID.
```

For a group or repeat group, use the id as listed in the editor or retrieved using the field Group Id or Repeat Id properties.

Once the Id of the object has been set, the property key must be set. The easiest way to find the key of a property is to look it up in Appendix F where all the properties are listed. The key is actually made up of a number of parts, defined in COBOL as follows:

```
10  QPR-PR-KEY.
    15  QPR-PR-OBJECT-TYPE  PIC X.
        88  QPR-PR-PANEL        VALUE "P".
        88  QPR-PR-STATIC       VALUE "S".
        88  QPR-PR-FIELD        VALUE "F".
        88  QPR-PR-GROUP        VALUE "G".
```

```
           88  QPR-PR-REPEAT       VALUE "R".
       15  QPR-PR-TYPE         PIC X.
           88  QPR-PR-LEN-T        VALUE "L".
           88  QPR-PR-NUM-T        VALUE "N".
           88  QPR-PR-CHAR-T       VALUE "C".
           88  QPR-PR-VAR-T        VALUE "V".
       15  QPR-PR-VAR-TYPE     PIC X.
           88  QPR-PR-VAR-1        VALUE "A".
           88  QPR-PR-VAR-2        VALUE "B".
           88  QPR-PR-VAR-3        VALUE "C".
           88  QPR-PR-VAR-4        VALUE "D".
           88  QPR-PR-VAR-5        VALUE "E".
           88  QPR-PR-VAR-6        VALUE "F".
           88  QPR-PR-VAR-7        VALUE "G".
           88  QPR-PR-VAR-8        VALUE "H".
           88  QPR-PR-VAR-9        VALUE "I".
           88  QPR-PR-VAR-10       VALUE "J".
       15  QPR-PR-OFFSET       PIC 9(5).
       15  QPR-PR-LEN          PIC 9(5).
       15  QPR-PR-FORMAT       PIC X.
           88  QPR-PR-NUMBER       VALUE "N".
           88  QPR-PR-BINARY       VALUE "B".
           88  QPR-PR-DECIMAL      VALUE "D".
       15  QPR-PR-ACTION       PIC X.
           88  QPR-PR-REDRAW       VALUE "R".
           88  QPR-PR-RECREATE     VALUE "C".
       15  QPR-PR-VAR-ACT      PIC X.
           88  QPR-PR-RESET-LEN VALUE "L".
```

By setting the values of these parts, it is possible to derive the value of the key itself without having to look it up. The parts are defined as follows:

Object type - identifies the object being accessed.
- P = panel
- S = static
- F = field
- G = group
- R = repeat

Type - identifies the section within the object definition (see qprmaint.cpy.)
- L = length
- N = num-data
- C = char-data
- V = var-data

Var-type - identifies the sub-section within the variable section of the object definition.
- A = first sub-section
- B = second sub-section
- C = third sub-section
- Etc.

Offset - the zero-based offset (in bytes) of the property within the section.

Length - the length (in bytes) of the property within the section.

Format - the format of the property value.
- Blank = character value held in pr-value
- N = numeric value held in pr-num-value for a numeric property
- B = binary value (00000000 thru 11111111) held in pr-bin-value for a character property, anything other than 0 or 1 means leave bit unchanged

D = decimal value (0-255) held in pr-num-value for a character property

Action - the action to be taken after the property has been set.
R = redraw object
C = recreate object

Var-act - the action to be taken if the property is in the variable portion of the object definition.
L = reset length of variable section to the length of the new value

For example, to reset the field Protection property in order to hide a field, use the following:

```
MOVE LOW-VALUES TO QPR-PR-DATA.
MOVE FIELD-ID TO QPR-PR-ID.
MOVE "FC-0003100001" TO QPR-PR-KEY.
MOVE "h" TO PR-VALUE.
CALL "QPR" USING QPR-SET-PROPERTY QPR-PROPERTY.
```

In this case:

Object type - F (field)
Type - C (char-data)
Var-type - - (N/A)
Offset - 00031
Length - 00001

CHAPTER C9

Dynamic Form Creation**Error! Bookmark not defined.**

As well as resetting an object's properties, as explained in the previous chapter, you can use the SET functions to actually create objects. This means that it is possible to define your forms without ever using the form editor. Remember that you must be in maintenance mode to use the SET functions and that the copy file QPRMAINT.CPY must be included in your program.

This chapter will not cover this topic in exhaustive detail for it is clearly a large topic. Instead, some basic concepts will be covered and then you should refer to the sample programs which illustrate the various techniques. Also refer to part E which fully describes all the properties that can be set when defining new objects.

The first thing to do is to obtain some memory in which to define a form. This is done using the OPEN-WINDOW function as follows:

```
MOVE LOW-VALUES TO QPR-WD-DATA.
MOVE "MEMORY" TO QPR-WD-NAME.
MOVE "y" TO QPR-WD-HIDE-SW.
CALL "QPR" USING QPR-OPEN-WINDOW QPR-WINDOW-DEF.
```

Once memory has been obtained, a form must be defined. This must be done before any fields can be defined.

```
MOVE LOW-VALUES TO QPR-PD-DATA.
MOVE 1 TO QPR-PD-CELL-WIDTH.
MOVE 1 TO QPR-PD-CELL-HEIGHT.
MOVE "FORM001" TO QPR-PD-NAME.
CALL "QPR" USING QPR-SET-PANEL-DEF QPR-PANEL-DEF.
```

Once a form has been defined, you can set up fields within the form. You will probably need to define both static and non-static fields. Static fields are used for general text items and field labels and for drawing lines. The following is an example:

```
MOVE LOW-VALUES TO QPR-SD-DATA.
MOVE 1 TO QPR-SD-ROW.
MOVE 10 TO QPR-SD-COL.
MOVE 200 TO QPR-SD-WIDTH.
MOVE 10 TO QPR-SD-HEIGHT.
MOVE "Some static text" TO QPR-SD-TEXT.
CALL "QPR" USING QPR-SET-STATIC-DEF QPR-STATIC-DEF.
```

The basic items that have to be set are position, size and the actual text. Notice that Row and Column are in terms of cells, and Width and Height are in terms of 1/10 cells. The size of a cell is defined at the form level.

Non-static fields are a little more complex. The following defines a custom field:

```
MOVE LOW-VALUES TO QPR-FD-DATA.
MOVE 1 TO QPR-FD-ID.
MOVE 5 TO QPR-FD-ROW.
MOVE 5 TO QPR-FD-COL.
MOVE 100 TO QPR-FD-WIDTH.
MOVE 0 TO QPR-FD-HEIGHT.
MOVE 10 TO QPR-FD-MAX-LEN.
MOVE 10 TO QPR-FD-PROG-LEN.
MOVE 0 TO QPR-FD-PROG-OFF.
MOVE 1 TO QPR-FD-PROG-NUM.
MOVE "l" TO QPR-FD-BOR-TYPE.
MOVE 0 TO QPR-FD-VAR-LEN.
MOVE LOW-VALUES TO QPR-FD-VAR-LENS.
CALL "QPR" USING QPR-SET-FIELD-DEF QPR-FIELD-DEF.
```

Here some extra items have to be set. Notice first that Height has been set to 0. This means that a default height should be used because it is hard to know what height to use for a field with a border.

All non-static fields have to have Id set. Here an id is being supplied by the program - if the FD-ID had been set to zero, an id would have been assigned automatically.

As well as having a physical width (the Width property), entry fields also have a width in terms of the number of characters that they can hold (the Maximum length property or FD-MAX-LEN item).

When you set up a custom field, you will normally want to be able to set the value of this field from your program. This is done by setting two more properties, Program length (FD-PROG-LEN) and Program offset (FD-PROG-OFF). Program length is the length of the data that is to be passed from the program and will normally be the same as Maximum length. Program offset is the offset of the data within your program's Field area. Normally, you will set this offset to 0 for the first field that you define and then increment it as you add more fields that will be receiving data from your program. It is important to make sure that you set this item correctly, otherwise you will not be able to set form data properly.

It is not necessary here to set any of the variable length field properties. The one that you may normally want to set is Format (FD-FMT) which holds the COBOL format of the field. If this format is not supplied the format defaults to PIC X for a length specified by Maximum length.

The default layout for FIELD-DEF defines the variable length properties area as a single PIC X item for a length of 2000. This definition allows needed flexibility because the makeup of the variable data area will be different depending upon the type of field that is being defined. To set up a value within this area use reference modification. The following sets a COBOL field format specifically:

```
MOVE "X(10)" TO QPR-FD-VAR-DATA (1 : 5).
MOVE 5 TO QPR-FD-FORMAT-LEN.
ADD 5 TO QPR-FD-VAR-LEN.
```

Once you have defined all your fields, your form is complete and can be used in a PRINT-PAGE function call. You do not have a generated converse-data area as you would have had if you had used the form editor to define the form but there is a generalized converse-data area defined within the QPRMAINT.CPY copy file and this can be used for processing any number of forms. You can use reference modification to move values in and out of the fields area based on Program offset (FD-PROG-OFF) and Program length (FD-PROG-LEN). You may have to modify the default size of CD-FIELD-DATA if you have a lot of variable fields in your form - if you do this, remember to reset CD-FIELD-LEN as well.

Once the values in the Fields area have been set, the form can be printed with a PRINT-PAGE call using the generalized converse-data area:

```
MOVE LOW-VALUES TO QPR-CD-DATA.
MOVE "FORM001" TO QPR-CD-NEXT-PANEL.
CALL "QPR" USING QPR-PRINT-PAGE QPR-CONVERSE-DATA.
```

CHAPTER C10
Form File Records in Working-Storage

Form file records can now be stored in working-storage eliminating the need for form files at runtime. The main disadvantage of this technique is that your programs will be bigger.

The working-storage definitions can be generated along with the regular cobol code using a new version of the code generator - use the template file qprnofil.cbx rather than qpr.cbx if you want to do this (see Appendix D for details on the code generator).

A new maintenance function, SET-RECORD, allows the records to be moved to Form Print memory so that they can be accessed by PRINT-PAGE as if they had just been read from a form file. For example:

```
CALL "QPR" USING QPR-ENTER-MAINT-MODE QPR-NULL-PARM.
CALL "QPR" USING QPR-SET-RECORD MYFORM-PANEL-RECORD.
MOVE LOW-VALUES TO QPR-WD-DATA.
MOVE "MYFORM" TO QPR-WD-PANEL-NAME.
MOVE "y" TO QPR-WD-HIDE-SW.
CALL "QPR" USING QPR-OPEN-WINDOW QPR-WINDOW-DEF.
CALL "QPR" USING QPR-EXIT-MAINT-MODE QPR-NULL-PARM.
```

Individual SET-RECORD calls must be made for each form but multiple PRINT-PAGE calls can be made using the same form without additional SET-RECORD calls.

The OPEN-WINDOW call is necessary to lock the form in memory otherwise it cannot be used in Print Preview mode. This memory window should be closed using the following code once the form is no longer needed:

```
CALL "QPR" USING QPR-ENTER-MAINT-MODE QPR-NULL-PARM.
MOVE "MYFORM" TO QPR-ND-NAME.
CALL "QPR" USING QPR-ACTIVATE-WINDOW QPR-NAME-DEF.
IF QPR-ND-RET-CODE = ZEROS
   CALL "QPR" USING QPR-CLOSE-WINDOW QPR-NULL-PARM
END-IF
CALL "QPR" USING QPR-EXIT-MAINT-MODE QPR-NULL-PARM.
```

PART D

COBOL FormPrint Programming Functions

CHAPTER D1
Introduction to Functions

This part of the manual explains the programming functions in detail. This chapter explains the various categories of functions that are available and the subsequent chapters define each function in detail.

Function calls are made to perform a specific task. Each function has an appropriate descriptive name and uses a single parameter which holds all data items needed by the function. This helps make your COBOL code easily understood and therefore easy to maintain.

Functions are classified into the following categories:

- Primary
- Mode switching
- Form
- Field
- Static field
- Group
- Repeat group
- Font
- Color
- Window
- File
- Miscellaneous

Primary functions are those which you will use in your program to print forms. The Mode switching functions are used to switch in and out of maintenance mode and the rest of the functions (apart from the About function) are used to modify objects at runtime and can only be used in maintenance mode.

Primary

INIT - initialize FormPrint and open a form file
SELECT-PRINTER(-EX) - select and configure a printer
PRINT-PAGE - print a form
END-PRINT - end print session

Mode switching

ENTER-MAINTENANCE-MODE - enter maintenance mode
EXIT-MAINTENANCE-MODE – return to regular mode

Form

GET-PANEL-DEF - get form properties
SET-PANEL-DEF - create form and/or set its properties
READ-NEXT-PANEL - get name of next form in current form file and read it into memory
WRITE-PANEL - write a form out from memory to current form file
SET-RECORD - set form record in memory

Field

GET-FIELD-DEF - get field properties
SET-FIELD-DEF - create field and/or set its properties
DELETE-FIELD - delete a field

GET-NEXT-FIELD-DEF - retrieve next field and get its properties

Static field

GET-STATIC-DEF - get static field properties
SET-STATIC-DEF - create static field and/or set its properties
DELETE-STATIC - delete a static field
GET-NEXT-STATIC-DEF - retrieve next static field and get its properties

Group

GET-GROUP-DEF - get group properties
SET-GROUP-DEF - create a group and/or set its properties
DELETE-GROUP - delete a group

Repeat group

GET-REPEAT-DEF - get repeat group properties
SET-REPEAT-DEF - create a repeat group and/or set its properties
DELETE-REPEAT - delete a repeat group

Font

GET-FONT-DEF - get font properties
SET-FONT-DEF - create a font and/or set its properties

Color

GET-COLOR-DEF - get color properties
SET-COLOR-DEF - create a color and/or set its properties

Window

OPEN-WINDOW - open a window
CLOSE-WINDOW - close a window
ACTIVATE-WINDOW - activate a window
CLEAR-WINDOW - clear a window

File

OPEN-FILE - open a form file
CLOSE-FILE - close a form file

Miscellaneous

ABOUT - get version information
GET-PROPERTY - get object property
SET-PROPERTY - set object property

Copy Files

All the functions and their parameters are defined in two copy files called QPR.CPY and QPRMAINT.CPY. You should include QPR.CPY in your program and you may optionally include QPRMAINT.CPY. If you have a program that is based on the code generated from the editor, QPR.CPY will automatically be included.

CHAPTER D2
Primary Functions

INIT

The INIT function initializes FormPrint and opens the form file containing the forms to be used for printing.

Parameter: INIT-AREA

Example:

```
MOVE LOW-VALUES TO QPR-QI-DATA.
MOVE "n" TO QPR-METHOD.
MOVE "testfile.pan" TO QPR-FILE.
CALL "QPR" USING QPR-INIT QPR-INIT-AREA.
```

SELECT-PRINTER(-EX)

The SELECT-PRINTER AND SELECT-PRINTER-EX functions are used to select and configure a printer. SELECT-PRINTER-EX must be used if you wish to set or access any of the extended properties.

Parameter: AREA.

Example:

```
MOVE LOW-VALUES TO QPR-AREA.
MOVE "y" TO QPR-DIALOG.
CALL "QPR" USING QPR-SELECT-PRINTER QPR-AREA.
```

PRINT-PAGE

The PRINT-PAGE function is used to print a form. Unless you are outputting multi-form pages, it will also cause a page to be printed. Multiple PRINT-PAGE calls will normally be made before an END-PRINT call (see below) is made.

Parameter: CONVERSE-DATA.

Example:

```
MOVE LOW-VALUES TO TESTFORM-DATA.
MOVE "TESTFORM" TO TESTFORM-NEXT-PANEL.
MOVE LOW-VALUES TO TESTFORM-FIELDS.
MOVE LOW-VALUES TO TESTFORM-COLRS.
CALL "QPR" USING QPR-PRINT-PAGE TESTFORM-CONVERSE-DATA.
```

END-PRINT

The END-PRINT function flushes out the pages to be printed and closes the form file. Once this function has been called, INIT and SELECT-PRINTER must be called before PRINT-PAGE can be used again.
Parameter: NULL-PARM.

Example:

```
CALL "QPR" USING QPR-END-PRINT QPR-NULL-PARM.
```

CHAPTER D3
Mode Switching Functions

ENTER-MAINTENANCE-MODE

The ENTER-MAINTENANCE-MODE function is used to switch modes so that the maintenance functions (Form, Field, Static Field, Group, Repeat Group, Color and Font) can be used to set and retrieve object properties. Once in maintenance mode, the primary printing functions cannot be used until an EXIT-MAINTENANCE-MODE call has been made.

Parameter: NULL-PARM

Example:

```
CALL "QPR" USING QPR-ENTER-MAINTENANCE-MODE QPR-NULL-PARM.
```

EXIT-MAINTENANCE-MODE

The EXIT-MAINTENANCE-MODE function is used to switch modes so that the primary printing functions can again be used. When in regular (printing) mode, only the primary functions can be used.

Parameter: NULL-PARM

Example:

```
CALL "QPR" USING QPR-EXIT-MAINTENANCE-MODE QPR-NULL-PARM.
```

CHAPTER D4
Form Functions

GET-PANEL-DEF

The GET-PANEL-DEF function is used to retrieve form properties. The key is PD-NAME.

Parameter: PANEL-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-PD-DATA.
MOVE "TESTFORM" TO QPR-PD-NAME.
CALL "QPR" USING QPR-GET-PANEL-DEF QPR-PANEL-DEF.
```

This retrieves the properties for the TESTFORM form. A window must be opened with TESTFORM in it before this code will work.

SET-PANEL-DEF

The SET-PANEL-DEF function is used to create a form and/or set its properties. The key is PD-NAME.

Parameter: PANEL-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-PD-DATA.
MOVE "TESTFORM" TO QPR-PD-NAME.
CALL "QPR" USING QPR-GET-PANEL-DEF QPR-PANEL-DEF.
MOVE 5 TO QPR-PD-FONT-ID.
CALL "QPR" USING QPR-SET-PANEL-DEF QPR-PANEL-DEF.
```

This retrieves a form's properties using GET-PANEL-DEF, sets the font, then updates the form properties. If you are modifying a form's properties you should always call GET-PANEL-DEF first.

READ-NEXT-PANEL

The READ-NEXT-PANEL function allows you to get the names of all the panels in a file. This is an easy way of applying changes to all forms without using the editor. Initialize the parameter to low-values in order to get the first form in the file. A non-zero return code signifies the end of the file.

Parameter: NAME-DEF

Example:

```
CALL "QPR" USING QPR-READ-NEXT-PANEL QPR-NAME-DEF.
IF QPR-ND-RET-CODE = 0
    MOVE LOW-VALUES TO QPR-PD-DATA
    MOVE QPR-ND-NAME TO QPR-PD-NAME
    CALL "QPR" USING QPR-GET-PANEL-DEF QPR-PANEL-DEF
    MOVE 10 TO QPR-PD-FONT-ID
    CALL "QPR" USING QPR-SET-PANEL-DEF QPR-PANEL-DEF
    CALL "QPR" USING QPR-WRITE-PANEL QPR-NULL-PARM.
```

This code could be used to update all the forms in a file.

WRITE-PANEL

The WRITE-PANEL function is used to write the current form definition out to the current file. The definition in this case includes all fields, static fields, groups, and other form properties:.

Parameter: NULL-PARM.

Example:

```
CALL "QPR" USING QPR-WRITE-PANEL QPR-NULL-PARM.
```

SET-RECORD

The SET-RECORD function is used to move a form definition from program working-storage to memory.

Parameter: PANEL-RECORD.

Example:

```
CALL "QPR" USING QPR-SET-RECORD MYFORM-RECORD.
```

This establishes MYFORM in memory based on the form definition produced by the generator and included in working-storage.

CHAPTER D5
Field Functions

GET-FIELD-DEF

The GET-FIELD-DEF function is used to retrieve a field's properties. The key is FD-ID or FD-NAME. If you wish to use FD-NAME, as the key, FD-ID should be set to 0.

Parameter: FIELD-DEF.

Example:

```
MOVE 2000 TO QPR-FD-VAR-LEN.
MOVE LOW—VALUES TO QPR-FD-VAR-LENS.
MOVE LOW-VALUES TO QPR-FD-DATA.
MOVE TESTFORM-NAME-FIELD-I TO QPR-FD-ID.
CALL "QPR" USING QPR-GET-FIELD-DEF QPR-FIELD-DEF.
```

This retrieves the properties of the field named NAME-FIELD in the TESTFORM form. The field ids are generated in the copy file for TESTFORM. Notice how FD-VAR-LEN and FD-VAR-LENS are initialized: FD-VAR-LEN should be set to the length of FD-VAR-DATA and all the other LEN items should be set to zero. If this is not done variable-length data items may be truncated.

SET-FIELD-DEF

The SET-FIELD-DEF function is used to create a field and/or set its properties. The key is FD-ID or FD-NAME. If you wish to use FD-NAME as the key, FD-ID should be set to 0.

Parameter: FIELD-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-FD-DATA.
MOVE TESTFORM-NAME-FIELD-I TO QPR-FD-ID.
CALL "QPR" USING QPR-GET-FIELD-DEF QPR-FIELD-DEF.
MOVE 5 TO QPR-FD-FONT-ID.
CALL "QPR" USING QPR-SET-FIELD-DEF QPR-FIELD-DEF.
```

This retrieves a field's properties using GET-FIELD-DEF, sets the font, then updates the field properties. If you are modifying field properties you should always call GET-FIELD-DEF first.

DELETE-FIELD

The DELETE-FIELD function is used to delete a field. The key is FD-ID.

Parameter: FIELD-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-FD-DATA.
MOVE TESTFORM-NAME-FIELD-I TO QPR-FD-ID.
CALL "QPR" USING QPR-DELETE-FIELD QPR-FIELD-DEF.
```

This deletes the name field on the TESTFORM form.

GET-NEXT-FIELD-DEF

The GET-NEXT-FIELD-DEF function allows you to sequentially retrieve fields and their properties.

Parameter: FIELD-DEF

Set FD-RET-CODE to -1 in order to start the retrieval process at the first field. 0 will be returned in FD-RET-CODE until all fields have been retrieved when 1 will be returned.

CHAPTER D6
Static Field Functions

GET-STATIC-DEF

The GET-STATIC-DEF is used to retrieve a static field's properties. The key is SD-ID or SD-ROW, SD-COL.

Parameter: STATIC-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-SD-DATA.
MOVE 1 TO QPR-SD-ROW.
MOVE 1 TO QPR-SD-COL.
CALL "QPR" USING QPR-GET-STATIC-DEF QPR-STATIC-DEF.
```

This retrieves the properties for the static at row 1, column 1. If you need to access static text on a regular basis it is better to make the static a display-only text field.

SET-STATIC-DEF

The SET-STATIC-DEF function is used to create a static field and/or set its properties. The key is SD-ID or SD-ROW, SD-COL.

Parameter: STATIC-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-FD-DATA.
MOVE 1 TO QPR-SD-ROW.
MOVE 1 TO QPR-SD-COL.
CALL "QPR" USING QPR-GET-STATIC-DEF QPR-STATIC-DEF.
MOVE "new text" TO QPR-SD-TEXT.
CALL "QPR" USING QPR-SET-STATIC-DEF QPR-STATIC-DEF.
```

This retrieves a static definition using GET-STATIC-DEF, modifies the static text, then updates the static definition. You should normally use a display-only text field to do this sort of processing.

DELETE-STATIC

The DELETE-STATIC function is used to delete a static field. The key is SD-ROW and SD-COL or SD-ID.

Parameter: STATIC-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-SD-DATA.
MOVE 1 TO QPR-SD-ROW.
MOVE 1 TO QPR-SD-COL.
CALL "QPR" USING QPR-DELETE-STATIC QPR-STATIC-DEF.
```

GET-NEXT-STATIC-DEF

The GET-NEXT-STATIC-DEF function allows you to sequentially retrieve static fields and their properties.

Parameter: STATIC-DEF

Set SD-RET-CODE to -1 in order to start the retrieval process at the first static field. 0 will be returned in SD-RET-CODE until all static fields have been retrieved when 1 will be returned.

Example:

```
MOVE -1 TO QPR-SD-RET-CODE.
PERFORM UNTIL QPR-SD-RET-CODE = 1
    MOVE LOW-VALUES TO QPR-SD-DATA
    CALL "QPR" USING QPR-GET-NEXT-STATIC-DEF QPR-STATIC-DEF
END-PERFORM.
```

CHAPTER D7
Group Functions

GET-GROUP-DEF

The GET-GROUP-DEF function is used to retrieve a group's properties. The key is GD-ID.

Parameter: GROUP-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-GD-DATA.
MOVE QPR-FD-GROUP-ID TO QPR-GD-ID.
CALL "QPR" USING QPR-GET-GROUP-DEF QPR-GROUP-DEF.
```

This retrieves the properties of a group containing a field whose properties have been previously retrieved using GET-FIELD-DEF.

SET-GROUP-DEF

The SET-GROUP-DEF function is used to create a group and/or set its properties. The key is GD-ID.

Parameter: GROUP-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-GD-DATA.
MOVE QPR-FD-GROUP-ID TO QPR-GD-ID.
CALL "QPR" USING QPR-GET-GROUP-DEF QPR-GROUP-DEF.
MOVE LOW-VALUE TO QPR-GD-SELECT-TYPE.
CALL "QPR" USING QPR-SET-GROUP-DEF QPR-GROUP-DEF.
```

This resets the Select type property which controls the functionality of the group. If you wish to make a change that affects the fields within the group you should delete the group then recreate it.

DELETE-GROUP

The DELETE-GROUP function is used to delete a group. The key is GD-ID.

Parameter: GROUP-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-GD-DATA.
MOVE QPR-FD-GROUP-ID TO QPR-GD-ID.
CALL "QPR" USING QPR-DELETE-GROUP QPR-GROUP-DEF.
```

This deletes the group containing a field whose properties have been previously retrieved.

CHAPTER D8
Repeat Group Functions

GET-REPEAT-DEF

The GET-REPEAT-DEF function is used to retrieve repeat group properties. The key is RD-ID.

Parameter: REPEAT-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-RD-DATA.
MOVE QPR-FD-REPEAT-ID TO QPR-RD-ID.
CALL "QPR" USING QPR-GET-REPEAT-DEF QPR-REPEAT-DEF.
```

This retrieves the properties for a repeat group whose properties have been previously retrieved using GET-FIELD-DEF.

SET-REPEAT-DEF

The SET-REPEAT-DEF function is used to create a repeat group and/or set its properties. The key is RD-ID.

Parameter: REPEAT-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-RD-DATA.
MOVE QPR-FD-REPEAT-ID TO QPR-RD-ID.
CALL "QPR" USING QPR-GET-REPEAT-DEF QPR-REPEAT-DEF.
CALL "QPR" USING QPR-DELETE-REPEAT QPR-REPEAT-DEF.
MOVE 200 TO QPR-RD-VERT-OCC.
CALL "QPR" USING QPR-SET-REPEAT-DEF QPR-REPEAT-DEF.
```

This modifies the total number of occurrences in a repeat group. If you wish to modify the number of occurrences you must delete the repeat group then recreate it.

DELETE-REPEAT

The DELETE-REPEAT function is used to delete a repeat group. The key is RD-ID.

Parameter: REPEAT-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-RD-DATA.
MOVE QPR-FD-REPEAT-ID TO QPR-RD-ID.
CALL "QPR" USING QPR-DELETE-REPEAT QPR-REPEAT-DEF.
```

This deletes the repeat group containing a field whose properties have been previously retrieved using GET-FIELD-DEF.

CHAPTER D9
Font Functions

GET-FONT-DEF

The GET-FONT-DEF function is used to retrieve font properties. The key is FO-ID.

Parameter: FONT-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-FO-DATA.
MOVE QPR-FD-FONT-ID TO QPR-FO-ID.
CALL "QPR" USING QPR-GET-FONT-DEF QPR-FONT-DEF.
```

This retrieves the font properties for a field whose properties have been previously retrieved using GET-FIELD-DEF.

SET-FONT-DEF

The SET-FONT-DEF function is used to create a font and/or set its properties. The key is FO-ID.

Parameter: FONT-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-FO-DATA.
MOVE 10 TO QPR-FO-WIDTH.
MOVE 20 TO QPR-FO-HEIGHT.
MOVE "Helv" TO QPR-FO-NAME.
CALL "QPR" USING QPR-SET-FONT-DEF QPR-FONT-DEF.
```

This creates a new Helvetica font. The font id will be placed in QPR-FO-ID upon return from the call.

CHAPTER D10
Color Functions

GET-COLOR-DEF

The GET-COLOR-DEF function is used to retrieve color properties. The key is CO-ID.

Parameter: COLOR-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-CO-DATA.
MOVE QPR-FD-COLR-ID TO QPR-CO-ID.
CALL "QPR" USING QPR-GET-COLOR-DEF QPR-COLOR-DEF.
```

This retrieves color properties for a field whose properties have been previously retrieved GET-FIELD-DEF.

SET-COLOR-DEF

The SET-COLOR-DEF function is used to create a color and/or set its properties. The key is CO-ID.

Parameter: COLOR-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-CO-DATA.
MOVE "t" TO QPR-CO-TYPE (1).
MOVE X"07" TO QPR-CO-TEXT (1).
MOVE "t" TO QPR-CO-TYPE (2).
MOVE X"01" TO QPR-CO-TEXT (2).
CALL "QPR" USING QPR-SET-COLOR-DEF QPR-COLOR-DEF.
```

This creates a color consisting of the text-mode shades of grey (low-intensity white) on blue. The color id will be placed in QPR-CO-ID upon return from the call.

CHAPTER D11
Window Functions

OPEN-WINDOW

The OPEN-WINDOW function opens a new window for maintaining forms. If WD-PANEL-NAME is set to a valid form name, that form will be automatically loaded into the new window named after the form.

Parameter: WINDOW-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-WD-DATA.
MOVE "TESTFORM" TO QPR-WD-PANEL-NAME.
CALL "QPR" USING QPR-OPEN-WINDOW QPR-WINDOW-DEF.
```

CLOSE-WINDOW

The CLOSE-WINDOW function closes a window. If you are using Print Preview, use ACTIVATE-WINDOW prior to CLOSE-WINDOW otherwise you may interfere with the Print Preview process.

Parameter: NULL-PARM.

Example:

```
CALL "QPR" USING QPR-CLOSE-WINDOW QPR-NULL-PARM.
```

This closes a window.

ACTIVATE-WINDOW

The ACTIVATE-WINDOW function makes a previously opened window the current window.

Parameter: NAME-DEF.

Example:

```
MOVE "TESTFORM" TO QPR-ND-NAME.
CALL "QPR" USING QPR-ACTIVATE-WINDOW QPR-NAME-DEF.
CALL "QPR" USING QPR-CLOSE-WINDOW QPR-NULL-PARM.
```

CLEAR-WINDOW

The CLEAR-WINDOW function clears a form out of a memory window.

Parameter: NULL-PARM

Example:

```
CALL "QPR" USING QPR-CLEAR-WINDOW QPR-NULL-PARM.
```

CHAPTER D12
File Functions

OPEN-FILE

The OPEN-FILE function opens the form file containing the forms to be used or created. A file is normally opened using the INIT function (see Chapter D2). The OPEN-FILE function allows you to open a file in read-only or update mode and would normally be used when maintaining forms for printing at some other time. In read-only mode, form definitions may still be updated temporarily (for example, using the SET-PANEL-DEF function) but the updates may not be written back to the file.

Parameter: FILE-DEF.

Example:

```
MOVE LOW-VALUES TO QPR-FI-DATA.
MOVE "testfile.pan" TO QPR-FI-NAME.
CALL "QPR" USING QPR-OPEN-FILE QPR-FILE-DEF.
```

CLOSE-FILE

The CLOSE-FILE function closes the current file. A file is normally closed using the END-PRINT function (see Chapter D2). The CLOSE-FILE function would normally be used when maintaining forms for printing at some other time.

Parameter: NULL-PARM.

Example:

```
CALL "QPR" USING QPR-CLOSE-FILE QPR-NULL-PARM.
```

CHAPTER D13
Miscellaneous Functions

ABOUT

The ABOUT function retrieves the version of FormPrint being used.

Parameter: VERSION

Example:

```
MOVE LOW-VALUES TO QPR-QE-DATA.
CALL "QPR" USING QPR-ABOUT QPR-VERSION.
```

GET-PROPERTY**Error! Bookmark not defined.**

The GET-PROPERTY function retrieves the value of a property for a form, field, etc.

Parameter: PROPERTY.

Example:

```
MOVE LOW-VALUES TO QPR-PR-DATA.
MOVE "PC-0000000008" TO QPR-PR-KEY.
CALL "QPR" USING QPR-GET-PROPERTY QPR-PROPERTY.
MOVE QPR-PR-VALUE TO CURRENT-FORM-NAME.
```

The above retrieves the name of the current form.

SET-PROPERTY

The SET-PROPERTY function sets the value of a property for a form, field, etc.

Parameter: PROPERTY.

Example:

```
MOVE LOW-VALUES TO QPR-PR-DATA.
MOVE PICTURE-I TO QPR-PR-ID.
MOVE "FVC0000800012" TO QPR-PR-KEY.
MOVE "MOUNTAIN.BMP" TO QPR-PR-VALUE.
CALL "QPR" USING QPR-SET-PROPERTY QPR-PROPERTY.
```

The above sets a new image to be printed.

PART E

COBOL FormPrint
Data Items

CHAPTER E1
Introduction to properties

Part E describes the properties that control the appearance and behavior of FormPrint objects (forms, fields, etc.). Most of these properties are set at design time using the form editor's properties box. Select the appropriate object and the properties of that object will be automatically displayed in the properties box. Many properties can be set by bringing up a list of possible values. These values are described in detail in this chapter.

Part E also lays out the parameters used at runtime to set these properties programmatically. In most cases the first 2 bytes of a parameter are reserved for a return code. This will be set to 0 unless an error occurs. Following the return code is a group of data items defining the lengths of the actual data to be passed or received. These lengths are very important and should not be changed unless you are specifically instructed to do so. Data items relating to actual properties are grouped together, depending upon whether they are numeric, character or of variable length. There is always a length item (as described above) relating to each group. The length items corresponding to any variable length data items may need to be reset to suit a particular situation.

The following is a list of the property types that are covered here:

- Initialization properties
- Printer properties
- Current form properties
- Form properties
- Field properties
- Static field properties
- Group properties
- Repeat group properties
- Font properties
- Color properties
- Window properties
- Version properties
- Miscellaneous parameters

CHAPTER E2
Initialization properties

Overview

These properties are used at runtime in order to initialize FormPrint and identify a forms file to be accessed.

The properties are set using data items in the QPR-INIT-AREA parameter on a QPR-INIT call. This parameter is contained in the copy file QPR.CPY.

Parameter layout

```
01  QPR-INIT-AREA.

    05 QPR-QI-RET            PIC S9(4)   COMP-5.
    05 QPR-QI-DATA.
       10  QPR-LIBRARY       PIC X.
       10  QPR-METHOD        PIC X.
       10  QPR-FILE          PIC X(80).
       10  FILLER            PIC X(118).
```

Library

Program name: **QPR-LIBRARY**

Must be set to low-value.

Method

Program name: **QPR-METHOD**

Must be set to "n".

File name

Program name: **QPR-FILE**

The name of the forms file to be accessed.

CHAPTER E3
Printer properties

Overview

These properties are used at runtime in order to select and configure a printer.

The properties are set using data items in the QPR-AREA parameter on a SELECT-PRINTER or SELECT-PRINTER-EX call. This parameter is contained in the copy file QPR.CPY. Properties marked with (*) can only be set and retrieved using the SELECT-PRINTER-EX call.

Parameter layout

```
01  QPR-AREA.

    05  QPR-RET-CODE                PIC S9(4)   COMP-5.
    05  QPR-DATA.
        10  QPR-DIALOG              PIC X.
        10  QPR-DOC-NAME            PIC X(30).
        10  QPR-ORIENTATION         PIC X.
        10  QPR-COPIES              PIC S9(4)   COMP-5.
        10  QPR-STRETCH             PIC X.
        10  QPR-DEVICE-LIST         PIC X(800).
        10  FILLER REDEFINES QPR-DEVICE-LIST.
            15  QPR-DEVICE-TO-SELECT PIC X(256).
            15  QPR-PRINT-FILE-NAME  PIC X(256).
        10  QPR-DRIVER              PIC X(32).
        10  QPR-OUTPUT              PIC X(32).
        10  QPR-DATA-EX.
            15  QPR-PAPER-SIZE      PIC X.
                88  QPR-LETTER                VALUE X"01".
                88  QPR-LEGAL                 VALUE X"05".
                88  QPR-A4                    VALUE X"09".
                88  QPR-B5                    VALUE X"0D".
            15  QPR-PAPER-SOURCE    PIC X.
                88  QPR-UPPER-TRAY            VALUE X"01".
                88  QPR-LOWER-TRAY            VALUE X"02".
                88  QPR-MIDDLE-TRAY           VALUE X"03".
                88  QPR-MANUAL-FEED           VALUE X"04".
            15  QPR-PREVIEW         PIC X.
                88  QPR-DO-PREVIEW            VALUE "y".
            15  QPR-LEFT-MARGIN     PIC S9(4) COMP-5.
            15  QPR-TOP-MARGIN      PIC S9(4) COMP-5.
            15  QPR-ALLOW-COPIES    PIC X.
                88  QPR-DO-COPIES             VALUE "y".
            15  QPR-DUPLEX          PIC X.
                88  QPR-DUP-SIMPLEX           VALUE X"01".
                88  QPR-DUP-VERTICAL          VALUE X"02".
                88  QPR-DUP-HORIZONTAL        VALUE X"03".
            15  FILLER              PIC X(8).
            15  QPR-PAPER-LENGTH    PIC S9(4) COMP-5.
            15  QPR-PAPER-WIDTH     PIC S9(4) COMP-5.
            15  QPR-PRINT-TO-FILE   PIC X.
                88  QPR-DO-FILE               VALUE "y".
            15  QPR-RANGE           PIC X.
                88  QPR-DO-RANGE              VALUE "y".
            15  QPR-RANGE-FROM
                                    PIC S9(4) COMP-5.
            15  QPR-RANGE-TO        PIC S9(4) COMP-5.
            15  QPR-HORZ-DPI        PIC S9(4) COMP-5.
            15  QPR-VERT-DPI        PIC S9(4) COMP-5.
```

```
           15  QPR-PHYS-LEFT-MARGIN
                                PIC S9(4) COMP-5.
           15  QPR-PHYS-TOP-MARGIN
                                PIC S9(4) COMP-5.

           15  FILLER           PIC X(67).
```

Dialog

Program name: **DIALOG**

Controls how the printer will be selected. Set to one of the following:

"y" - initiates Windows Printer Dialog.
"s" - returns all available printers (if QPR-RET-CODE = 0) or selects a specific printer (if QPR-RET-CODE = -10).
"d" or low-value - selects default printer.
"a" - displays Printer Dialog after exiting Print Preview but before actual printing starts. This is particularly useful in connection with the Range facility - see Range property below. If not in Preview mode, this setting has the same effect as "y".

Document name

Program name: **DOC-NAME**

The name of the document to be shown in the print queue.

Orientation

Program name: **ORIENTATION**

The orientation of the paper. Set to one of the following:

"p" or low-value - Portrait.
"l" (lower-case "L") - Landscape.

Copies

Program name: **COPIES**

The number of copies to print (zero = 1 copy). More than one copy will be generated only if the printer driver supports the automatic generation of extra copies. If the Dialog property is set to "y", the user will normally only be able to request additional copies if the driver supports the automatic generation of extra copies. This behavior may be changed, however, using the Allow copies property (see below). If Allow copies is not set, Copies will be reset to reflect the number of copies to be automatically generated by the printer.

Stretch

Program name: **STRETCH**

Not used.

Device list

Program name: **DEVICE-LIST**

The list of available printers. Each printer will be separated by a semicolon (;). For example:

HP LaserJet 4p/4MP;HP DeskJet 510; Microline 183.....

This list is available on return from a SELECT-PRINTER function call with DIALOG set to "s". RET-CODE will also be set to -10.

Device to select

Program name: **DEVICE-TO-SELECT**

The specific printer to be selected. DIALOG must be set to "s" and RET-CODE must be set to -10. If this value is set when DIALOG is "y", it is used as the default printer to be displayed in the printer dialog box.

Print file name (*)

Program name: **PRINT-FILE-NAME**

The name of the print file to be generated. The Print to file property (see below) must be set to "y".

Driver

Program name: **DRIVER**

The name of the driver being used. Available on return from a SELECT-PRINTER function call.

Output

Program name: **OUTPUT**

The name of the port being used. Available on return from a SELECT-PRINTER function call.

Paper size (*)

Program name: **PAPER-SIZE**

The size of the paper to be used. Set to one of the following:

Low-value - default
X"01" - letter
X"05" - legal
X"09" - A4
X"0d" - B5

Paper source (*)

Program name: **PAPER-SOURCE**

The source of the paper being used. This is set to one of the following by the runtime:

X"01" - upper tray
X"02" - lower tray
X"03" - middle tray
X"04" - manual feed

Preview (*)

Program name: **PREVIEW**

Switch to indicate that output should be previewed before sending it to the printer. Set to one of the following:

Low-value - no preview
y - enter preview mode
d - enter preview mode and allow for dynamically modified forms and multi-form pages

Left margin (*)

Program name: **LEFT-MARGIN**

The size of the left margin in 1/1000 inches. This allows you to start printing at a precise position on the page regardless of the printer being used so it is useful in conjunction with preprinted forms. If the value is zero or less than the size of the printer hardware margin, the printer hardware margin will be used.

Top margin (*)

Program name: **TOP-MARGIN**

The size of the top margin in 1/1000 inches. This allows you to start printing at a precise position on the page regardless of the printer being used so it is useful in conjunction with preprinted forms. If the value is zero or less than the size of the printer hardware margin, the printer hardware margin will be used.

Allow copies (*)

Program name: **ALLOW-COPIES**

Switch to allow the user to enter the number of copies to be printed even if the printer driver does not support the automatic generation of extra copies. If this property is set to "y", the Copies property will be reset upon return from a SELECT-PRINTER-EX call to reflect the number of copies that must be generated by the program i.e. if the COPIES data item is greater than 1, the PRINT-PAGE should be called multiple times for each page in order to generate the copies.

Duplex (*)

Program name: **DUPLEX**

Specifies that double-sided printing should be activated if the printer supports it. Set to one of the following:

X"01" or low-value - single-sided
X"02" - vertical double-sided
X"03" - horizontal double sided

Paper length (*)

Program name: **PAPER-LENGTH**

The height of the page in 1/10 millimeters (zero = default length). This property would normally only be used in conjunction with continuous-feed printers where the page size is variable. If Paper length and Paper width are set to -1 prior to a SELECT-PRINTER-EX call, they will be reset to reflect the printable area on the page (in 1/10 millimeters). Use the formula (x * 96) / 254 to convert either of these values into logical pixels.

Paper width (*)

Program name: **PAPER-WIDTH**

The width of the page in 1/10 millimeters (zero = default width). This property would normally only be used in conjunction with continuous-feed printers where the page size is variable. If Paper length and Paper width are set to -1 prior to a SELECT-PRINTER-EX call, they will be reset to reflect the printable area on the page (in 1/10 millimeters). Use the formula (x * 96) / 254 to convert either of these values into logical pixels.

Print to file (*)

Program name: **PRINT-TO-FILE**

Switch (if set to "y") to specify that print output should be directed to a file rather than the printer itself. This file may be copied to the printer at a later time. The user will be prompted for the name of the file unless the Print file name property is set (see above).

Range (*)

Program name: **RANGE**

Switch (if set to "y") to specify that the user should be able to select a range of pages to be printed. Also set Range-from to the lowest page number that can be entered - normally 1. Set Range-to to the highest page number that can be entered - this must be estimated if the exact number of pages is not known. These two numbers will be updated to reflect the numbers actually chosen by the user. Range will be reset to low-value if the user chooses to print all pages. The PRINT-PAGE function will check to see if the current page is within the range selected and suppress output if not. If thecurrent page is past the selected range, -21 will be returned in form-KEY to indicate that the remainder of the report need not be generated and END-PRINT may be called immediately.

Range-from (*)

Program name: **RANGE-FROM**

The lowest page number that may be entered by the user - normally 1. Updated by the SELECT-PRINTER-EX function to reflect the range actually chosen by the user.

Range-to (*)

Program name: **RANGE-TO**

The highest page number that may be entered by the user - this must be estimated if the exact number of pages is not known. Updated by the SELECT-PRINTER-EX function to reflect the range actually chosen by the user.

Horizontal DPI (*)

Program name: **HORZ-DPI**

The horizontal resolution (dots per inch) of the printer. Value is returned by SELECT-PRINTER-EX call.

Vertical DPI (*)

Program name: **HORZ-DPI**

The horizontal resolution (dots per inch) of the printer. Value is returned by SELECT-PRINTER-EX call.

Physical Left Margin (*)

Program name: **PHYS-LEFT-MARGIN**

The left hardware margin 1/1000 inches. Value is returned by SELECT-PRINTER-EX call.

Physical Top Margin (*)

Program name: **PHYS-LEFT-MARGIN**

The top hardware margin 1/1000 inches. Value is returned by SELECT-PRINTER-EX call.

CHAPTER E4
Current Form Properties

Overview

These properties are used at runtime in order to control the appearance and processing of a form. Some of the properties are used to override actual form and field properties set at design time.

The properties are set and accessed using data items in the CONVERSE-DATA parameter before a PRINT-PAGE function call. This parameter is contained in an individual copy file generated using the code generator. The version of CONVERSE-DATA included in QPRMAINT.CPY should only be used if forms are being defined dynamically.

Parameter Layout

(xxxxxxxx represents the name of a form):

```
  01  XXXXXXXX-CONVERSE-DATA.
      05  XXXXXXXX-RET-CODE          PIC S9(4) COMP-5.
      05  XXXXXXXX-LENS.
          10  XXXXXXXX-LEN-LEN       PIC S9(4) COMP-5 VALUE +20.
          10  XXXXXXXX-IP-NUM-LEN    PIC S9(4) COMP-5 VALUE +40.
          10  XXXXXXXX-IP-CHAR-LEN   PIC S9(4) COMP-5 VALUE +106.
          10  XXXXXXXX-OP-NUM-LEN    PIC S9(4) COMP-5 VALUE +6.
          10  XXXXXXXX-OP-CHAR-LEN   PIC S9(4) COMP-5 VALUE +2.
          10  XXXXXXXX-FIELD-LEN     PIC S9(4) COMP-5 VALUE +0.
          10  XXXXXXXX-COLR-LEN      PIC S9(4) COMP-5 VALUE +0.
          10  FILLER                 PIC S9(4) COMP-5 VALUE +0.
          10  FILLER                 PIC S9(4) COMP-5 VALUE +0.
          10  FILLER                 PIC S9(4) COMP-5 VALUE +0.
      05  XXXXXXXX-DATA.
 ******** XXXXXXXX-IP-NUM-DATA *****
          10  XXXXXXXX-KEY           PIC S9(4) COMP-5.
          10  FILLER                 PIC X(38).
******** XXXXXXXX-IP-CHAR-DATA ****
          10  XXXXXXXX-NEXT-PANEL    PIC X(8).
          10  FILLER                 PIC X(94).
          10  XXXXXXXX-WAIT-SW       PIC X.
          10  FILLER                 PIC X(3).
******** XXXXXXXX-OP-NUM-DATA *****
          10  XXXXXXXX-PAN-POS-SW    PIC S9(4) COMP-5.
          10  XXXXXXXX-PAN-ROW       PIC S9(4) COMP-5.
          10  XXXXXXXX-PAN-COL       PIC S9(4) COMP-5.
 ******** XXXXXXXX-OP-CHAR-DATA ****
          10  FILLER                 PIC X(2).
 ******** XXXXXXXX-FIELDS *******
 ******** XXXXXXXX-COLRS ********
```

xxxxxxxx-FIELD-LEN is the length of xxxxxxxx-FIELDS.
xxxxxxxx-COLR-LEN is the length of xxxxxxxx-COLRS.

Control key pressed

Program name : xxxxxxxx-KEY

The special action that caused control to be returned to your program during Print Preview. May be set to one of the following values:

-17 = "Cancel print" has been selected

-18 = "Exit without printing" has been selected
-24 = "Exit with printing" has been selected

Next form

Program name: **xxxxxxxx-NEXT-PANEL**

The form to be processed.

Wait switch

Program name: **xxxxxxxx-WAIT-SW**

Switch used to indicate multi-form page processing.

n = do not print page yet but wait for another form to be processed.

Form position switch

Program name: **xxxxxxxx-PAN-POS-SW**

Switch used to control positioning of current form on a multi-form page.

1 = use Form row and column properties to position form on page

Form row

Program name: **xxxxxxxx-PAN-ROW**

The row for the form in terms of cells.

Form column

Program name: **xxxxxxxx-PAN-COL**

The column for the form in terms of cells.

Fields area

Program name: **xxxxxxxx-FIELDS**

Data items used to override the Initial value property of each field in the form that has its Program length property set to non-zero. If a data item is set to low-values, the Initial value of the field will be displayed, otherwise the value of the data item will be displayed.

Program length is usually derived from the field Format property at design time and the code generator will generate data items based on this. The order of the data items is dependent on the field Program offset property which is usually calculated by the Editor based on the order of fields in the form and their length.

Colors area

Program name: **xxxxxxxx-COLRS**

Data items used to override the color property of each field in the form that has its Program number property set to non-zero. If a data item is set to low-values, the color of the field will be unchanged.

The order of the data items is dependent on the field Program number property which is usually calculated by the Editor based on the order of the fields in the form.

Each data item can be set to a hexadecimal number representing a numeric color code, as follows:

low-value = no change

x'0B' or gtr = user-defined color

CHAPTER E5
Form properties

Overview

These properties define the appearance of a form.

Form properties can be set and accessed programmatically in maintenance mode either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the PANEL-DEF parameter before a SET-PANEL-DEF call and after a GET-PANEL-DEF call. To set a property using SET-PANEL-DEF after a panel is already defined you would normally make a GET-PANEL-DEF call followed by a SET-PANEL-DEF call. The PANEL-DEF parameter is contained in the copy file QPRMAINT.CPY.

Parameter layout

```
 01  QPR-PANEL-DEF.
     05  QPR-PD-RET-CODE         PIC S9(4) COMP-5.
     05  QPR-PD-LENS.
         10  QPR-PD-LEN-LEN      PIC S9(4) COMP-5 VALUE +24.
         10  QPR-PD-NUM-LEN      PIC S9(4) COMP-5 VALUE +40.
         10  QPR-PD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +76.
         10  QPR-PD-VAR-LEN      PIC S9(4) COMP-5 VALUE +132.
         10  QPR-PD-DESC-LEN     PIC S9(4) COMP-5 VALUE +20.
         10  QPR-PD-TITLE-LEN    PIC S9(4) COMP-5 VALUE +20.
         10  FILLER              PIC S9(4) COMP-5 VALUE +50.
         10  FILLER              PIC S9(4) COMP-5 VALUE +40.
         10  FILLER              PIC S9(4) COMP-5 VALUE +0.
         10  FILLER              PIC S9(4) COMP-5 VALUE +1.
         10  FILLER              PIC S9(4) COMP-5 VALUE +1.
         10  FILLER              PIC S9(4) COMP-5 VALUE +1.
     05  QPR-PD-DATA.
******** QPR-PD-NUM-DATA ********
         10  FILLER              PIC X(8).
         10  QPR-PD-FLD-CNT      PIC S9(4) COMP-5.
         10  QPR-PD-PROG-CNT     PIC S9(4) COMP-5.
         10  QPR-PD-PROG-LEN     PIC S9(4) COMP-5.
         10  FILLER              PIC X(14).
         10  QPR-PD-TOT-WIDTH    PIC S9(4) COMP-5.
         10  QPR-PD-TOT-HEIGHT   PIC S9(4) COMP-5.
         10  FILLER              PIC X(2).
         10  QPR-PD-CELL-WIDTH   PIC S9(4) COMP-5.
         10  QPR-PD-CELL-HEIGHT  PIC S9(4) COMP-5.
         10  QPR-PD-FONT-ID      PIC S9(4) COMP-5.
******** QPR-PD-CHAR-DATA *******
         10  QPR-PD-NAME         PIC X(8).
         10  FILLER              PIC X(16).
         10  QPR-PD-FORMAT-NUMS  PIC X.
         10  FILLER              PIC X.
         10  QPR-PD-INIT-NUMS    PIC X.
         10  FILLER              PIC X(3).
         10  QPR-PD-MISC-OPTIONS PIC X.
         10  FILLER              PIC X(7).
         10  QPR-PD-COLR         PIC X.
         10  QPR-PD-PROG-DATE    PIC X.
         10  FILLER              PIC X(26).
         10  QPR-PD-MORE-OPTIONS PIC X.
         10  FILLER              PIC X(9).
******** QPR-PD-VAR-DATA ********
         10  QPR-PD-DESCRIPTION  PIC X(20).
         10  QPR-PD-TITLE        PIC X(20).
```

```
10  FILLER              PIC X(50).
10  FILLER              PIC X(40).
10  FILLER              PIC X.
10  FILLER              PIC X.
10  FILLER              PIC X.
```

PD-DESC-LEN is the length of PD-DESCRIPTION.
PD-TITLE-LEN is the length of PD-TITLE.

Field count

Program name: **PD-FLD-CNT**

The count of fields in the form. This property is maintained automatically as fields are added and deleted.

Program field count

Program name: **PD-PROG-CNT**

The count of fields accessible by your program. This property is maintained automatically as fields are added and deleted.

Program field length

Program name: **PD-PROG-LEN**

The length of fields accessible by your program. This property is maintained automatically as fields are added and deleted.

Total width

Program name: **PD-TOT-WIDTH**

The total width of the form in cells.

Total height

Program name: **PD-TOT-HEIGHT**

The total height of the form in cells.

Cell width

Program name: **PD-CELL-WIDTH**

The width of a cell in pixels.

Cell height

Program name: **PD-CELL-HEIGHT**

The height of a cell in pixels.

Font ID

Program name: **PD-FONT-ID**

The id of the font to be used for text in the form (unless overridden at the field level). Fonts can be established in the form editor or by using the SET-FONT-DEF function. Zero means use the default font.

Name

Program name: **PD-NAME**

The name of the form. Set in the form editor using the File/Save or File/Save as menu bar option. If a form is assigned a name that cannot be used as part of a COBOL data name, a warning message is issued. Set PD-NAME before a SET-PANEL-DEF or GET-PANEL-DEF call to indicate which form is to be accessed.

Format numerics

Program name: **PD-FORMAT-NUMS**

Used to control whether program numeric values should be reformatted i.e. they should be assumed to be COBOL de-edited numeric items with an assumed decimal point and sign combined with the last digit.

Low-value = reformat numerics
n = do not reformat

Initialize numerics

Program name: **PD-INIT-NUMS**

Used to control whether numeric fields are initialized to zero.

low-value = initialize numerics
n = numeric field will be initialized to blanks.

Cell height increment

Program name: **PD-CELL-HT-INC**

Allows finer control of line spacing and vertical print output. For example, if you want to achieve exactly 10 lines per inch, set Cell height to 9 and Cell height increment to 60. This will give you 96 / 9.60 = 10 lpi when you print. Cell height increment has no effect on screen output.

Miscellaneous

Program name: **PD-MISC-OPTIONS**

Switch used to control various form appearance. The following value can be set:

X'80' - allow icon fields to be overlaid with other fields.

Color

Program name: **PD-COLR**

The id of the color of the form. This only effects objects within the form, not the form background.

low-value = use default color

Program date format

Program name: **PD-PROG-DATE**

The format of dates in your program's Fields area.

d = DMY
m = MDY
low-value = YMD
'/' (slash)= M/D/Y
Separators are normally removed to leave a length of either 6 or 8, depending on whether the date includes the century.

More options

Program name: **PD-MORE-OPTIONS**

A switch controlling form appearance. The following value may be set:

X'02' - total size (Total width and height properties) is specified in 1/100 inches

Description

Program name: **PD-DESCRIPTION**

Descriptive text for the form.

Title

Program name: **PD-TITLE**

The title of the form to be displayed in the title bar of the Print Preview window.

CHAPTER E6
Field properties

Overview

These properties define the appearance of a field.

Field properties are set and accessed programmatically in maintenance mode either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the FIELD-DEF parameter before a SET-FIELD-DEF call and after a GET-FIELD-DEF call. To set a property using SET-FIELD-DEF after a field is already defined you would normally make a GET-FIELD-DEF call followed by a SET-FIELD-DEF call. The FIELD-DEF parameter is contained in the copy file QPRMAINT.CPY.

Parameter layout

```
 01  QPR-FIELD-DEF.
     05  QPR-FD-RET-CODE         PIC S9(4) COMP-5.
     05  QPR-FD-LENS.
         10  QPR-FD-LEN-LEN      PIC S9(4) COMP-5 VALUE +26.
         10  QPR-FD-NUM-LEN      PIC S9(4) COMP-5 VALUE +44.
         10  QPR-FD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +74.
         10  QPR-FD-VAR-LEN      PIC S9(4) COMP-5 VALUE +2000.
         10  QPR-FD-VAR-LENS.
             15  QPR-FD-FORMAT-LEN
                                 PIC S9(4) COMP-5 VALUE +0.
             15  QPR-FD-CAPTION-LEN
                                 PIC S9(4) COMP-5 VALUE +0.
             15  QPR-FD-INITIAL-LEN
                                 PIC S9(4) COMP-5 VALUE +0.
             15  FILLER          PIC S9(4) COMP-5 VALUE +0.
             15  FILLER
                                 PIC S9(4) COMP-5 VALUE +0.
             15  QPR-FD-DISCRETE-LEN
                                 PIC S9(4) COMP-5 VALUE +0.
             15  FILLER
                                 PIC S9(4) COMP-5 VALUE +0.
             15  FILLER          PIC S9(4) COMP-5 VALUE +0.
             15  FILLER          PIC S9(4) COMP-5 VALUE +0.
     05  QPR-FD-DATA.
******** QPR-FD-NUM-DATA ********
         10  QPR-FD-ID           PIC S9(4) COMP-5.
         10  FILLER              PIC X(4).
         10  QPR-FD-OCCURRENCE   PIC S9(4) COMP-5.
         10  QPR-FD-BASE-ID      PIC S9(4) COMP-5.
         10  QPR-FD-ROW          PIC S9(4) COMP-5.
         10  QPR-FD-COL          PIC S9(4) COMP-5.
         10  QPR-FD-PROG-OFF     PIC S9(4) COMP-5.
         10  FILLER              PIC X(4).
         10  QPR-FD-PROG-NUM     PIC S9(4) COMP-5.
         10  QPR-FD-WIDTH        PIC S9(4) COMP-5.
         10  QPR-FD-HEIGHT       PIC S9(4) COMP-5.
         10  QPR-FD-MAX-LEN      PIC S9(4) COMP-5.
         10  QPR-FD-PROG-LEN     PIC S9(4) COMP-5.
         10  QPR-FD-ITEM-LEN     PIC S9(4) COMP-5.
         10  FILLER              PIC X(6).
         10  QPR-FD-GROUP-ID     PIC S9(4) COMP-5.
         10  QPR-FD-REPEAT-ID    PIC S9(4) COMP-5.
         10  QPR-FD-FONT-ID      PIC S9(4) COMP-5.
******** QPR-FD-CHAR-DATA *******
         10  QPR-FD-NAME         PIC X(30).
```

```
           10  QPR-FD-TYPE          PIC X.
           10  QPR-FD-PROG-DEC      PIC X.
           10  FILLER               PIC X(4).
           10  QPR-FD-INIT-NUMS     PIC X.
           10  FILLER               PIC X(20).
           10  QPR-FD-JUSTIFY       PIC X.
           10  QPR-FD-FILL          PIC X.
           10  FILLER               PIC X.
           10  QPR-FD-SPEC-FMT      PIC X.
           10  FILLER               PIC X(3).
           10  QPR-FD-CURS-SKIP     PIC X.
           10  FILLER               PIC X(2).
           10  QPR-FD-BLANK-ZERO    PIC X.
           10  QPR-FD-CTRL-TYPE     PIC X.
           10  QPR-FD-COLR          PIC X.
           10  FILLER               PIC X.
           10  QPR-FD-BOR-TYPE      PIC X.
           10  QPR-FD-PROG-SPEC     PIC X.
******** QPR-FD-VAR-DATA ********
           10  QPR-FD-VAR-DATA      PIC X(2000).
******** QPR-FD-FMT *************
******** QPR-FD-CAPTION *********
******** QPR-FD-INITIAL-VAL *****
******** QPR-FD-DISC-VALS *******
```

FD-FORMAT-LEN is the length of FD-FMT.
FD-CAPTION-LEN is the length of FD-CAPTION.
FD-INITIAL-LEN is the length of FD-INITIAL-VAL.
FD-DISCRETE-LEN is the length of FD-DISC-VALS.

Id

Program name: **FD-ID**

The id used to identify the field. Set automatically in the form editor when a new field is defined. If FD-ID is set to zero before a SET-FIELD-DEF call, a new field will be created with an automatically assigned id. In order to find out the id used in this case, set FD-ID to -1 and make a GET-FIELD-DEF call and on return FD-ID will be reset to the id used. Otherwise, FD-ID should be set to the id of the field to be accessed by GET-FIELD-DEF or SET-FIELD-DEF.

Occurrence

Program name: **FD-OCCURRENCE**

The occurrence number of the field if it is part of a repeat group. Repeat fields are generated automatically when repeat groups are defined so field definitions with an occurrence number greater than 1 should not be accessed directly.

Base id

Program name: **FD-BASE-ID**

The id of the original field if this is a repeat field. Repeat fields are generated automatically when repeat groups are defined so field definitions with an occurrence number greater than 1 should not be accessed directly.

Row

Program name: **FD-ROW**

The row in cells within the form where the field is positioned. May be set directly in the form editor or indirectly by selecting and dragging the field.

Column

Program name: **FD-COL**

The column in cells within the form where the field is positioned. May be set directly in the form editor or indirectly by selecting and dragging the field.

Program offset

Program name: **FD-PROG-OFF**

The offset within the program Fields area of the field. Automatically set in the form editor prior to code generation or saving the form.

Program number

Program name: **FD-PROG-NUM**

The occurrence number of the field within in the program Colors area (see Current form properties). Automatically set by the form editor prior to code generation or saving the form.

Width

Program name: **FD-WIDTH**

The width of the field in terms of 1/10 cells. May be set directly in the form editor or indirectly by selecting the field and dragging the left or right border.

Height

Program name: **FD-HEIGHT**

The height of the field in terms of 1/10 cells. May be set directly in the form editor or indirectly by selecting the field and dragging the top or bottom border. May be set to 0 which means use a default height for the field.

Maximum length

Program name: **FD-MAX-LEN**

The maximum number of characters that can be displayed in the field. Only required for custom fields (Control type = low-value). Automatically set in the form editor based on the Format property.

Program length

Program name: **FD-PROG-LEN**

The length of the field as it appears in the program Fields area. Automatically set in the form editor based on the Format property.

Item length

Program name: **FD-ITEM-LEN**

For an Icon field (Control type = "i"), if Item length is not zero it specifies the length of the name of the icon file in the Value property (see Value property). Automatically set in the editor based on the Format property. This must be set as well as Program length in order to use the program Fields area to change an icon at runtime.

Group id

Program name: **FD-GROUP-ID**

The id of the group to which this field belongs. Set automatically when the group is defined.

Repeat id

Program name: **FD-REPEAT-ID**

The id of the repeat group to which this field belongs. Set automatically when the repeat group is defined.

Font id

Program name: **FD-FONT-ID**

The id of the font to be used for text in the field. Fonts can be established in the form editor or by using the SET-FONT-DEF function. Zero means use the default font.

Name

Program name: **FD-NAME**

The name of the field.

Type

Program name: **FD-TYPE**

For a custom field (Control type = low-value), Type controls the sort of data that will be displayed in the field:

low-value = any data
n = numeric
d = date

For an icon field (Control type = "i"), Type specifies the type of "icon":

low-value = regular bitmap (file name is in the Initial value property)
X"01" = bitmap-exact. This setting is used to display 256-color bitmaps using the exact colors that were used when the bitmap was created. This is usually necessary when displaying photographs, for instance. It is not advisable to use this setting if you need to display more than one bitmap within the same form.
X"05" = Activex control. Field will act as a container for the activex control as identified by the Value property.

Protection

Program name: **FD-OUTPUT**

Protection controls whether a field is visible. The default is visible. This property can only be reset at runtime.

h = hidden
other = visible

Decimals

Program name: **FD-PROG-DEC**

Specifies the number of decimals in the program's definition of this field. This allows the program definition to be independent of the display definition. This property is only referenced if the Program Data property (FD-PROG-SPEC) is set appropriately.

Initialize if numeric

Program name: **FD-INIT-NUMS**

Specifies if a numeric field should be initialized to zero.

low-value = default to form Initialize numerics property
n = do not initialize to zero

Miscellaneous

Program name: **FD-MISC-OPTIONS**

Controls various aspects of field behavior. The following values are combined to form the final value:

X'40' = the text associated with this field will be treated as an escape sequence and sent directly to the printer. For example, the sequence:
\1b&a10C\1b&a20Rhello\0d\0a
causes "hello" to be printed in the default font at column 10, row 20 on an HP printer.

Justify

Program name: **FD-JUSTIFY**

Controls whether data will be justified in any way before it is displayed within the field. Right justification is useful for lining up numeric fields in neat columns.

low-value = no justification
l = justify to the left
r = justify to the right
c = center

Fill character

Program name: **FD-FILL**

Specifies the character that will be used to fill blank spaces after user input.

low-value = zero for numeric fields, blank for non-numeric fields.

Special format

Program name: **FD-SPEC-FMT**

For a custom field (Control type property is low-value), specifies that the field format contains special separator characters. Only X, 9 and A in the field format will be treated as a variable character slot. Do not use parentheses to specify multiple X's, 9's or A's in the field format.

low-value = regular format
y = special format

For a bitmap icon field (Control type property is "i" and Type property is low-value or X"01"), specifies if the field or image is to be automatically resized.

low-value = clip image to field size
i = resize field to image size
f = resize bitmap image to field size

Usage option

Program name: **FD-CURS-SKIP**

For a custom field (Control type property is low-value), specifies a single- or multi-line field.

low-value = single-line field
m = multi-line field

Blank if zero

Program name: **FD-BLANK-ZERO**

Controls whether a numeric field should be displayed as blank if its value is zero.

low-value = do not display blank
y = blank if zero

Control type

Program name: **FD-CTRL-TYPE**

The type of the field. Normally set in the form editor by creating a new field with one of the field icons in the toolbar.

low-value = Custom field
r = Radio Button
c = Check Box
i = Icon

Color

Program name: **FD-COLR**

The id of the color of the field.

low-value = refer to form Color property

Border type

Program name: **FD-BOR-TYPE**

For custom fields (Control type property is low-value), specifies the type of border for the field.

low-value = no border
l (lower case "L") = default border
3 = old-style 3-dimensional border
t = thin 3d border

Program data

Program name: **FD-PROG-SPEC**

Controls whether the Program offset, Program length, Program number and Decimals properties can be specified manually rather than be calculated automatically. The following values are combined to form the final value:

low-value = calculate these properties automatically
X'01' = set Program offset property
X'02' = set Program length property
X'04' = set Program number property
X'08' = set Decimals property

Format

Program name: **FD-FMT**

The COBOL format of the field i.e. display format. This may be a non-standard format if the Special format property is set to "y". If the field is a Date Field, use:

'm' for month

'd' for day
'y' for year

For example, mm/dd/yy would be used for the format of Month/Day/Year.

Caption

Program name: **FD-CAPTION**

The caption for a Radio Button or Check Box.

Value

Program name: **FD-INITIAL-VAL**

The data to be displayed in the field if not set by your program.

For a date field (Type property is "d"), this property can be set to "tt/tt/tt" (set FD-INITIAL-VAL to "tttttt" in your program) to indicate that today's date should be automatically displayed in the field.

For an icon field (Control type property is "i"), this property is set as follows:

Bytes 9-20 = icon file (eg. "myimage.bmp").

If Item length is set for an Icon field then the name of the icon file starts at the first byte of the Value property for a length of Item length.

For an activex control icon (Control type property is "i" and Type property is X"05"):

Bytes 21 - = used to store the activex control name and property data in the form "name/event/event/property=value/property=value/etc.".

Discrete values

Program name: **FD-DISC-VALS**

For a Radio Button, the list should be just one value which is the value to be used to turn the Radio Button on. For a Check Box, the list should be two items, the first value being the value to be used to check the Check Box, the second to uncheck it.

CHAPTER E7
Static field properties

Overview

These properties define the appearance of a static field.

Static field properties are set and accessed programmatically in maintenance mode either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the STATIC-DEF parameter before a SET-STATIC-DEF call and after a GET-STATIC-DEF call. To set a property using SET-STATIC-DEF after a static field is already defined you would normally make a GET-STATIC-DEF call followed by a SET-STATIC-DEF call. The STATIC-DEF parameter is contained in the copy file QPRMAINT.CPY.

Parameter layout

```
01  QPR-STATIC-DEF.
      05  QPR-SD-RET-CODE         PIC S9(4) COMP-5.
      05  QPR-SD-LENS.
          10  QPR-SD-LEN-LEN      PIC S9(4) COMP-5 VALUE +10.
          10  QPR-SD-NUM-LEN      PIC S9(4) COMP-5 VALUE +12.
          10  QPR-SD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +4.
          10  QPR-SD-VAR-LEN      PIC S9(4) COMP-5 VALUE +80.
          10  QPR-SD-TEXT-LEN     PIC S9(4) COMP-5 VALUE +80.
      05  QPR-SD-DATA.
 ******** QPR-SD-NUM-DATA ********
          10  QPR-SD-ID           PIC S9(4) COMP-5.
          10  QPR-SD-ROW          PIC S9(4) COMP-5.
          10  QPR-SD-COL          PIC S9(4) COMP-5.
          10  QPR-SD-WIDTH        PIC S9(4) COMP-5.
          10  QPR-SD-HEIGHT       PIC S9(4) COMP-5.
          10  QPR-SD-FONT-ID      PIC S9(4) COMP-5.
 ******** QPR-SD-CHAR-DATA *******
          10  QPR-SD-COLR         PIC X.
          10  QPR-SD-TYPE         PIC X.
          10  QPR-SD-JUSTIFY      PIC X.
          10  QPR-SD-MISC-OPTIONS PIC X.
 ******** QPR-SD-VAR-DATA ********
          10  QPR-SD-TEXT         PIC X(80).
```

SD-TEXT-LEN is the length of SD-TEXT.

Id

Program name: **SD-ID**

The id for the static field. The form editor uses the Row and Column properties to index static fields so this property is usually set to zero. Setting this property manually however, may make it easier to access a static field programmatically using GET-STATIC-DEF and SET-STATIC-DEF.

Row

Program name: **SD-ROW**

The row in cells within the form where the static field is positioned. This property may be set directly in the form editor or indirectly by selecting and dragging the static field.

Column

Program name: **SD-COLUMN**

The column in cells within the form where the static field is positioned. This property may be set directly in the form editor or indirectly by selecting and dragging the static field.

Width

Program name: **SD-WIDTH**

The width of the static field in 1/10 cells. This property may be set directly in the form editor or indirectly by selecting and resizing the static field. If SD-WIDTH is set to 0 on a SET-STATIC-DEF call, Width will be calculated based on the value of the Text property.

Height

Program name: **SD-HEIGHT**

The height of the static field in 1/10 cells. This property may be set directly in the form editor or indirectly by selecting and resizing the static field. If this property is set to 0, a default height will be used for single-line text; for multi-line text, the height will be calculated to match the text.

Font ID

Program name: **SD-FONT-ID**

The id of the font to be used for static text. Fonts can be established in the form editor or by using the SET-FONT-DEF function. Zero means use the form font.

Color

Program name: **SD-COLR**

The id of the color of the field. Colors can be established in the form editor or by using the SET-COLOR-DEF function.

low-value = use form color (form Color property)

Type

Program name: **SD-TYPE**

The type of the static item. It can be one of the following:

low-value = regular text

'a' = aligned text. This causes the text to be positioned as if it were contained in a rectangular border, allowing the text to line up with an adjacent text field with a border.

'l' = line. This causes a line to be drawn. If the Width property is less than the Height property, the line will be vertical for a length of Height; if the Height property is less than the Width property, the line will be horizontal for a length of Width. The first byte of the Text property must be set to one of the following:

X'DA' = top left corner
X'C4' = horizontal
X'BF' = top right corner
X'B3' = vertical
X'C0' = bottom left
X'D9' = bottom right
X'C3' = left-T
X'B4' = right-T
X'C2' = top-T
X'C1' = bottom-T

X'C5' = cross

'm' = Multi-line text with word-wrap

Justify

Program name: **SD-JUSTIFY**

Specifies how text will be justified within a static. Right justified text is useful for right-aligning a series of labels adjacent to a corresponding series of left-aligned text fields.

low-value = no justification
l = justify to the left
r = justify to the right
c = center

Miscellaneous

Program name: **SD-MISC-OPTIONS**

Controls various aspects of a static field's appearance. The following values are combined to form the final value:

X'01' = 3d Line type
X'40' = the text associated with this field will be treated as an escape sequence and sent directly to the printer. For example, the sequence:
\1b&a10C\1b&a20Rhello\0d\0a
causes "hello" to be printed in the default font at column 10, row 20 on an HP printer.

Text

Program name: **SD-TEXT**

The text to be displayed in the static field. If the Type property is set to "l" (line), the first byte of the Text property should be set to one of the following:

X'DA' = top left corner
X'C4' = horizontal
X'BF' = top right corner
X'B3' = vertical
X'C0' = bottom left
X'D9' = bottom right
X'C3' = left-T
X'B4' = right-T
X'C2' = top-T
X'C1' = bottom-T
X'C5' = cross

CHAPTER E8
Group properties

Overview

These properties define the behavior of a group.

Group properties are set and accessed programmatically in maintenance mode either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the GROUP-DEF parameter before a SET-GROUP-DEF call and after a GET-GROUP-DEF call. To set a property using SET-GROUP-DEF after a group is already defined you would normally make a GET-GROUP-DEF call followed by a SET-GROUP-DEF call. The GROUP-DEF parameter is contained in the copy file QPRMAINT.CPY.

Parameter layout

```
 01  QPR-GROUP-DEF.
     05  QPR-GD-RET-CODE          PIC S9(4) COMP-5.
     05  QPR-GD-LENS.
         10  QPR-GD-LEN-LEN       PIC S9(4) COMP-5 VALUE +8.
         10  QPR-GD-NUM-LEN       PIC S9(4) COMP-5 VALUE +12.
         10  QPR-GD-CHAR-LEN      PIC S9(4) COMP-5 VALUE +48.
         10  QPR-GD-VAR-LEN       PIC S9(4) COMP-5 VALUE +20.
     05  QPR-GD-DATA.
******** QPR-GD-NUM-DATA ********
         10  QPR-GD-ID            PIC S9(4) COMP-5.
         10  QPR-GD-ROW           PIC S9(4) COMP-5.
         10  QPR-GD-COL           PIC S9(4) COMP-5.
         10  QPR-GD-WIDTH         PIC S9(4) COMP-5.
         10  QPR-GD-HEIGHT        PIC S9(4) COMP-5.
         10  QPR-GD-FLD-CNT       PIC S9(4) COMP-5.
******** QPR-GD-CHAR-DATA *******
         10  QPR-GD-NAME          PIC X(30).
         10  FILLER               PIC X(3).
         10  QPR-GD-SELECT-TYPE   PIC X.
         10  FILLER               PIC X(10).
         10  QPR-GD-COLR          PIC X.
         10  FILLER               PIC X.
         10  QPR-GD-BOR-TYPE      PIC X.
         10  FILLER               PIC X.
******** QPR-GD-VAR-DATA ********
         10  QPR-GD-FLD-ID OCCURS 10
                                  PIC S9(4) COMP-5.
```

Id

Program name: **GD-ID**

The id of the group. Automatically set in the form editor when a new group is defined. If GD-ID is set to zero before a SET-GROUP-DEF call, a new group will be created with an automatically assigned id. Otherwise, GD-ID should be set to the id of the group to be accessed by GET-GROUP-DEF or SET-GROUP-DEF.

Row

Program name: **GD-ROW**

The row in cells where the top of the group is located within the form. May be set directly in the form editor or indirectly by selecting and then dragging the group. The row does not allow for the group border (if present) which is drawn outside the actual group.

Column

Program name: **GD-COL**

The column in cells where the left of the group is located within the form. May be set directly in the form editor or indirectly by selecting and then dragging the group. The column does not allow for the group border (if present) which is drawn outside the actual group.

Width

Program name: **GD-WIDTH**

The width of the group in 1/10 cells. May be set directly in the form editor or by selecting and dragging the left or right hand border of the group. The width does not allow for the group border (if present) which is drawn outside the actual group.

Height

Program name: **GD-HEIGHT**

The height of the group in 1/10 cells. May be set directly in the form editor or by selecting and dragging the top or bottom border of the group. The height does not allow for the group border (if present) which is drawn outside the actual group.

Field count

Program name: **GD-FLD-CNT**

The number of fields included in the group. Set automatically by the form editor when the group is sized based on the number of fields within the area occupied by the group.

Name

Program name: **GD-NAME**

The name of the group.

Select type

Program name: **GD-SELECT-TYPE**

Specifies the functionality of the group.

low-value =	no special action
s =	single-select, only one field in the group may be selected (used for Radio Buttons)
m =	multiple-select, multiple fields in the group may be selected (used for Check Boxes)
n =	no member, used for drawing boxes without having any effect on fields and other groups lying within the boundaries of the box.

Color

Program name: **GD-COLR**

The id of the color of the group background.

low-value = use form color (form Color property)

Border type

Program name: **GD-BOR-TYPE**

The border to be drawn around the group.

low-value = no border

l (lower-case L) =	default border
3 =	3d border
t =	thin 3d border.
P =	system 3d group border
a =	single line border
A =	thick line border
b =	raised border
B =	raised and thick
c =	lowered border
C =	lowered and thick
d =	engraved border
D =	engraved and thick
e =	rimmed border
E =	rimmed and thick

Field ids

Program name: **GD-FLD-ID**

The ids of the fields included in the group. Set automatically by the form editor when the group is sized based on the fields within the area occupied by the group.

CHAPTER E9
Repeat group properties

Overview

These properties define the appearance of a repeat group.

Repeat group properties are set and accessed programmatically in maintenance mode either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the REPEAT-DEF parameter before a SET-REPEAT-DEF call and after a GET-REPEAT-DEF call. To set a property using SET-REPEAT-DEF after a group is already defined you would normally make a GET-REPEAT-DEF call followed by a SET-REPEAT-DEF call. The REPEAT-DEF parameter is contained in the copy file QPRMAINT.CPY.

Parameter layout

```
 01  QPR-REPEAT-DEF.
     05  QPR-RD-RET-CODE         PIC S9(4) COMP-5.
     05  QPR-RD-LENS.
         10  QPR-RD-LEN-LEN      PIC S9(4) COMP-5 VALUE +10.
         10  QPR-RD-NUM-LEN      PIC S9(4) COMP-5 VALUE +44.
         10  QPR-RD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +4.
         10  QPR-RD-VAR-LEN      PIC S9(4) COMP-5 VALUE +104.
         10  QPR-RD-BASE-LEN     PIC S9(4) COMP-5 VALUE +20.
     05  QPR-RD-DATA.
******** QPR-RD-NUM-DATA ********
         10  QPR-RD-ID           PIC S9(4) COMP-5.
         10  QPR-RD-ROW          PIC S9(4) COMP-5.
         10  QPR-RD-COL          PIC S9(4) COMP-5.
         10  QPR-RD-WIDTH        PIC S9(4) COMP-5.
         10  QPR-RD-HEIGHT       PIC S9(4) COMP-5.
         10  QPR-RD-VERT-OCC     PIC S9(4) COMP-5.
         10  FILLER              PIC S9(4) COMP-5.
         10  QPR-RD-VERT-GAP     PIC S9(4) COMP-5.
         10  FILLER              PIC X(6).
         10  QPR-RD-HOR-OCC      PIC S9(4) COMP-5.
         10  FILLER              PIC S9(4) COMP-5.
         10  QPR-RD-HOR-GAP      PIC S9(4) COMP-5.
         10  FILLER              PIC X(12).
         10  QPR-RD-PROG-LEN     PIC S9(4) COMP-5.
         10  QPR-RD-FLD-CNT      PIC S9(4) COMP-5.
******** QPR-RD-CHAR-DATA *******
         10  QPR-RD-BOR-TYPE     PIC X.
         10  FILLER              PIC X.
         10  QPR-RD-MISC-OPTIONS PIC X.
         10  FILLER              PIC X.
******** QPR-RD-VAR-DATA ********
         10  QPR-RD-BASE-ID OCCURS 10
                                 PIC S9(4) COMP-5.
```

RD-BASE-LEN is the length of all occurrences of RD-BASE-ID.

Id

Program name: **RD-ID**

The id of the repeat group. Automatically set in the form editor when a new repeat group is defined. If RD-ID is set to zero before a SET-REPEAT-DEF call, a new repeat group will be created with an automatically assigned id. Otherwise, RD-ID should be set to the id of the repeat group to be accessed by GET-GROUP-DEF or SET-GROUP-DEF.

Row

Program name: **RD-ROW**

The row in cells within the form where the top of the repeat group is located. May be set directly in the form editor or indirectly by selecting and dragging the repeat group. The row does not allow for the border (if present) which is drawn outside the actual repeat group.

Column

Program name: **RD-COL**

The column in cells within the form where the left of the repeat group is located. May be set directly in the form editor or indirectly by selecting and dragging the repeat group. The column does not allow for the border (if present) which is drawn outside the actual repeat group.

Width

Program name: **RD-WIDTH**

The width of the repeat group in 1/10 cells. May be set directly in the form editor or indirectly by selecting and then dragging the left or right hand border of the repeat group. The width does not allow for the border (if present) which is drawn outside the actual repeat group.

Height

Program name: **RD-HEIGHT**

The height of the repeat group in 1/10 cells. May be set directly in the form editor or indirectly by selecting and then dragging the top or bottom border of the repeat group. The height does not allow for the border (if present) which is drawn outside the actual repeat group.

Vertical occurs

Program name: **RD-VERT-OCC**

The total number of vertical occurrences within the repeat group.

Vertical gap

Program name: **RD-VERT-GAP**

The vertical gap between the occurrences. The unit for this gap is normally the number of cells (vertically) occupied by the first field in the repeat group so that the row incrementor is calculated as (field Height / 10) * (Vertical gap + 1). This unit may be changed to single cells by setting the repeat group Miscellaneous property so that the row incrementor is calculated as (field Height / 10 + Vertical gap).

The default gap is zero (consecutive rows) but this may be adjusted automatically if the fields within the repeat group are positioned on different rows.

Horizontal occurrences

Program name: **RD-HOR-OCC**

The total number of horizontal occurrences in the repeat group. There is no check to see that all the occurrences fit within the bounds of the repeat group.

Horizontal gap

Program name: **RD-HOR-GAP**

The number of cells between horizontal occurrences in the repeat group. If this is set to zero, the occurrences will be adjacent to each other.

Program field length

Program name: **RD-PROG-LEN**

The total length of the fields in a single vertical occurrence based on the Program length property of each field. Automatically set when a repeat group is defined.

Field count

Program name: **RD-FLD-CNT**

The number of base fields included in the repeat group. Set automatically by the form editor when the repeat group is sized based on the number of fields within the area occupied by the repeat group.

Border type

Program name: **RD-BOR-TYPE**

The border to be drawn around the repeat group:

low-value =	no border
l (lower-case L) =	default border
3 =	3-dimensional border
t =	thin 3d border.

Miscellaneous

Program name: **RD-MISC-OPTIONS**

Controls repeat group appearance.

X'01' -	vertical gap in cells (see Vertical gap property)

Field ids

Program name: **RD-BASE-ID**

The ids of the fields included in the repeat group. Set automatically by the form editor when the repeat group is sized based on the fields within the area occupied by the group.

CHAPTER E10
Font properties

Overview

These properties define the appearance of a font.

Font properties are set in the form editor by displaying the font selection list for a form, field or static field Font property and then clicking the right mouse button or pressing F2. A new font is defined using the standard system dialog box but an existing font is changed using a custom dialog box that allows more direct access to the font properties.

Font properties are set and accessed programmatically in maintenance mode using data items in the FONT-DEF parameter before a SET-FONT-DEF call and after a GET-FONT-DEF call. To set a property after a font is already defined you would normally make a GET-FONT-DEF call followed by a SET-FONT-DEF call. The FONT-DEF parameter is contained in the copy file QPRMAINT.CPY.

Parameter layout

```
01  QPR-FONT-DEF.
      05  QPR-FO-RET-CODE         PIC S9(4) COMP-5.
      05  QPR-FO-LENS.
          10  QPR-FO-LEN-LEN      PIC S9(4) COMP-5 VALUE +8.
          10  QPR-FO-NUM-LEN      PIC S9(4) COMP-5 VALUE +14.
          10  QPR-FO-CHAR-LEN     PIC S9(4) COMP-5 VALUE +6.
          10  QPR-FO-VAR-LEN      PIC S9(4) COMP-5 VALUE +30.
      05  QPR-FO-DATA.
 ******** QPR-FO-NUM-DATA ********
          10  QPR-FO-ID           PIC S9(4) COMP-5.
          10  QPR-FO-WIDTH        PIC S9(4) COMP-5.
          10  QPR-FO-HEIGHT       PIC S9(4) COMP-5.
          10  FILLER              PIC X(6).
          10  QPR-FO-WIDTH-INC    PIC S9(4) COMP-5.
 ******** QPR-FO-CHAR-DATA *******
          10  QPR-FO-PITCH        PIC X.
          10  QPR-FO-WEIGHT       PIC X.
          10  QPR-FO-ITALIC       PIC X.
          10  QPR-FO-STRIKE-OUT   PIC X.
          10  QPR-FO-UNDERLINE    PIC X.
          10  QPR-FO-CHAR-SET     PIC X.
 ******** QPR-FO-VAR-DATA ********
          10  QPR-FO-NAME         PIC X(30).
```

Id

Program name: **FO-ID**

The id of the font. Set in the form editor when a new font is defined (see Overview above). If FO-ID is set to zero before a SET-FONT-DEF call, a new font will be created with an automatically assigned id. Otherwise, FO-ID should be set to the id of the font to be accessed by GET-FONT-DEF or SET-FONT-DEF.

Width

Program name: **FO-WIDTH**

The average width of a character in the font in pixels. Set in the form editor based on the point size of the font defined (see Overview above).

Height

Program name: **FO-HEIGHT**

The height of a character in the font in pixels. Set in the form editor based on the point size of the font defined (see Overview above).

Width increment

Program name: **FO-WIDTH-INC**

Allows finer control of printer font width. This is important because printers typically have a much higher resolution (more dots per inch) than video devices. In FormPrint, screen dpi is fixed at 96 pixels. If you want 16 characters per inch, that's fine because you can use a font of 6 (96 / 16 = 6). However, if you want 17 cpi, that's a problem because you can't specify a width with pixel fractions (96 / 17 = 5.6) so you would have to use a width of 5 which would actually give you 19 cpi (96 / 5 = 19.2). Font width increment allows you to specify a pixel fraction for the printer even though this increment cannot be rendered on the screen. For 17 cpi, therefore, you could specify a width of 5 and an increment of 60, which would cause a printer font width of 35 dots to be used on a 600 dpi printer (5.60 * 600 / 96 = 35). See also QPRFON x3 configuration setting in Appendix A.

Pitch

Program name: **FO-PITCH**

The pitch of the font. Set in the form editor when a new font is defined (see Overview above).

low-value = variable pitch
f = fixed pitch

Weight

Program name: **FO-WEIGHT**

The weight of the font. Set in the form editor when a new font is defined (see Overview above).

low-value = normal
b = bold

Italic

Program name: **FO-ITALIC**

Specifies whether the font is italicized. Set in the form editor when a new font is defined (see Overview above).

low-value = normal
y = italic

Strike out

Program name: **FO-STRIKE-OUT**

Specifies whether the font is struck out with a horizontal line. Set in the form editor when a new font is defined (see Overview above).

low-value = normal
y = struck out

Underline

Program name: **FO-UNDERLINE**

Specifies whether the font is underlined. Set in the form editor when a new font is defined (see Overview above).

low-value = normal
y = underlined

Character set

Program name: **FO-CHAR-SET**

Specifies the character set used by the font. Set in the form editor when a new font is defined (see Overview above).

low-value = ANSI character set

Examples of non-ANSI fonts are the symbol fonts and also some international fonts. If a font is non-ANSI, it will be flagged on the font list dialog box in the "Ch" (non-ANSI Character set) column.

If you are using QPRIBM=1 (see appendix A), do not use characters outside the 32-127 range when using a non-ANSI font.

Name

Program name: **FO-CHAR-SET**

The name of the font type face such as "Helvetica", "Times Roman", or "Courier". Set in the form editor when a new font is defined (see Overview above).

CHAPTER E11
Color properties

Overview

These properties define the appearance of a color.

Color properties are set in the form editor by displaying the color selection list for a form, field or static field Color property and then clicking the right mouse button or pressing F2.

Color properties are set and accessed programmatically in maintenance mode using data items in the COLOR-DEF parameter before a SET-COLOR-DEF call and after a GET-COLOR-DEF call. To set a property after a font is already defined you would normally make a GET-COLOR-DEF call followed by a SET-COLOR-DEF call. The COLOR-DEF parameter is contained in the copy file QPRMAINT.CPY.

Parameter layout

```
 01  QPR-COLOR-DEF.
     05  QPR-CO-RET-CODE         PIC S9(4) COMP-5.
     05  QPR-CO-LENS.
         10  QPR-CO-LEN-LEN      PIC S9(4) COMP-5 VALUE +8.
         10  QPR-CO-NUM-LEN      PIC S9(4) COMP-5 VALUE +2.
         10  QPR-CO-CHAR-LEN     PIC S9(4) COMP-5 VALUE +16.
         10  QPR-CO-VAR-LEN      PIC S9(4) COMP-5 VALUE +30.
     05  QPR-CO-DATA.
******** QPR-CO-NUM-DATA ********
         10  QPR-CO-ID           PIC S9(4) COMP-5.
******** QPR-CO-CHAR-DATA *******
         10  QPR-CO-FG-BG OCCURS 2.
             15  QPR-CO-NUM      PIC X.
             15  QPR-CO-TYPE     PIC X.
             15  QPR-CO-SYSTEM   PIC X.
             15  QPR-CO-TEXT     PIC X.
             15  QPR-CO-RED      PIC X.
             15  QPR-CO-GREEN    PIC X.
             15  QPR-CO-BLUE     PIC X.
             15  FILLER          PIC X.
******** QPR-CO-VAR-DATA ********
         10  QPR-CO-NAME         PIC X(30).
```

Id

Program name: **CO-ID**

The id of the color. Set in the form editor when a new color is defined (see Overview above). If CO-ID is set to zero before a SET-COLOR-DEF call, a new color will be created with an automatically assigned id. Otherwise, CO-ID should be set to the id of the color to be accessed by GET-COLOR-DEF or SET-COLOR-DEF.

Number

Program name: **CO-NUM**

Used internally. The first occurrence of color properties refers to the foreground color, the second occurrence refers to the background color.

Type

Program name: **CO-TYPE**

The type of color being used. Set in the form editor when a new color is defined (see Overview above).

low-value = standard system color
t = equivalent text-mode color
r = defined using red, green and blue values

System color

Program name: **CO-SYSTEM**

The system color being used, if the Type property is low-value. Set in the form editor when a new color is defined (see Overview above).

Low-value = default color
x'01' = default text
x'02' = default window
x'03' = highlight text
x'04' = highlight
x'05' = menu text
x'06' = menu
x'07' = button text
x'08' = button
x'09' = button highlight
x'0A' = button shadow
x'0B' = grayed text
x'0C' = transparent background

Text color

Program name: **CO-TEXT**

The text color being used, if the Type property is "t". Set in the form editor when a new color is defined (see Overview above).

X'00' =	Black	X'08' =	dark grey
X'01' =	Blue	X'09' =	light blue
X'02' =	green	X'0a' =	light green
X'03' =	cyan	X'0b' =	light cyan
X'04' =	red	X'0c' =	pink
X'05' =	magenta	X'0d' =	light magenta
X'06' =	brown	X'0e' =	yellow
X'07' =	grey	X'0f' =	white

Red content

Program name: **CO-RED**

The red content of the color, if the Type property is "r". The value of the Red property can range from X'00' to X'ff'. Set in the form editor when a new color is defined (see Overview above).

Green content

Program name: **CO-GREEN**

The green content of the color, if the Type property is "r". The value of the Green property can range from X'00' to X'ff'. Set in the form editor when a new color is defined (see Overview above).

Blue content

Program name: **CO-BLUE**

The blue content of the color, if the Type property is "r". The value of the Blue property can range from X'00' to X'ff'. Set in the form editor when a new color is defined (see Overview above).

Name

Program name: **CO-NAME**

The name of the color. This makes RGB colors easier to identify in the color selection list in the form editor. Set in the form editor when a new color is defined (see Overview above).

CHAPTER E12
Window properties

Overview

These properties define the attributes of a window. A window is a memory-based object used as a container for a form while it is being created or modified.

Window properties are set using data items in the WINDOW-DEF parameter before an OPEN-WINDOW call. The WINDOW-DEF parameter is contained in the copy file QPRMAINT.CPY.

Parameter layout

```
 01  QPR-WINDOW-DEF.
     05  QPR-WD-RET-CODE          PIC S9(4) COMP-5.
     05  QPR-WD-LENS.
         10  QPR-WD-LEN-LEN       PIC S9(4) COMP-5 VALUE +10.
         10  QPR-WD-NUM-LEN       PIC S9(4) COMP-5 VALUE +38.
         10  QPR-WD-CHAR-LEN      PIC S9(4) COMP-5 VALUE +38.
         10  QPR-WD-VAR-LEN       PIC S9(4) COMP-5 VALUE +80.
         10  FILLER               PIC S9(4) COMP-5 VALUE +80.
     05  QPR-WD-DATA.
******** QPR-WD-NUM-DATA ********
         10  FILLER               PIC X(38).
******** QPR-WD-CHAR-DATA *******
         10  QPR-WD-NAME          PIC X(8).
         10  QPR-WD-PANEL-NAME    PIC X(8).
         10  FILLER               PIC X(13).
         10  QPR-WD-HIDE-SW       PIC X.
         10  FILLER               PIC X(8).
******** QPR-WD-VAR-DATA *******
         10  FILLER               PIC X(80).
```

WD-VAR-LEN is the length of WD-TITLE.
WD-TITLE-LEN is the length of WD-TITLE.

Name

Program name: **WD-NAME**
Derived from: **Form Name property**

The name of the window.

Form name

Program name: **WD-PANEL-NAME**

The name of the current form within the window. On an OPEN-WINDOW call, you may initialize WD-DATA to low-values and set WD-PANEL-NAME to indicate the name of the form to be loaded into memory.

Hide switch

Program name: **WD-HIDE-SW**

Must be set to "y".

CHAPTER E13
Version properties

Overview

These properties allow version information to be accessed and displayed. They are used in conjunction with the ABOUT function.

Parameter layout

```
01  QPR-VERSION.
    05  QPR-QE-RET                    PIC S9(4)   COMP-5.
      05  QPR-QE-DATA.
          10  QPR-VERSION-INFO        PIC X(80).
          10  QPR-DISPLAY-BOX         PIC X.
```

Version number

Program name: **VERSION-INFO**

The version of FormPrint being used (as a string).

Display messagebox

Program name: **DISPLAY-BOX**

Switch to specify that version information should be displayed in a messagebox:

Space or low-value - display messagebox
Other - return value of Version number property

CHAPTER E14
Property parameter

Overview

The PROPERTY parameter is used in conjunction with the GET-PROPERTY and SET-PROPERTY functions.

Parameter layout

```
01  QPR-PROPERTY.
    05  QPR-PR-RET-CODE         PIC S9(4) COMP-5.
    05  QPR-PR-LENS.
        10  QPR-PR-LEN-LEN      PIC S9(4) COMP-5 VALUE +18.
        10  QPR-PR-NUM-LEN      PIC S9(4) COMP-5 VALUE +6.
        10  QPR-PR-CHAR-LEN     PIC S9(4) COMP-5 VALUE +20.
        10  QPR-PR-VAR-LEN      PIC S9(4) COMP-5 VALUE +2000.
    05  QPR-PR-DATA.
******* QPR-PR-NUM-DATA ********
        10  QPR-PR-ID           PIC S9(4) COMP-5.
        10  QPR-PR-ROW          PIC S9(4) COMP-5.
        10  QPR-PR-COL          PIC S9(4) COMP-5.
******* QPR-PR-CHAR-DATA *******
        10  QPR-PR-KEY.
            15  QPR-PR-OBJECT-TYPE
                                PIC X.
                88  QPR-PR-PANEL    VALUE "P".
                88  QPR-PR-STATIC   VALUE "S".
                88  QPR-PR-FIELD    VALUE "F".
                88  QPR-PR-GROUP    VALUE "G".
                88  QPR-PR-REPEAT   VALUE "R".
            15  QPR-PR-TYPE     PIC X.
                88  QPR-PR-LEN-T    VALUE "L".
                88  QPR-PR-NUM-T    VALUE "N".
                88  QPR-PR-CHAR-T   VALUE "C".
                88  QPR-PR-VAR-T    VALUE "V".
            15  QPR-PR-VAR-TYPE PIC X.
                88  QPR-PR-VAR-1    VALUE "A".
                88  QPR-PR-VAR-2    VALUE "B".
                88  QPR-PR-VAR-3    VALUE "C".
                88  QPR-PR-VAR-4    VALUE "D".
                88  QPR-PR-VAR-5    VALUE "E".
                88  QPR-PR-VAR-6    VALUE "F".
                88  QPR-PR-VAR-7    VALUE "G".
                88  QPR-PR-VAR-8    VALUE "H".
                88  QPR-PR-VAR-9    VALUE "I".
                88  QPR-PR-VAR-10   VALUE "J".
            15  QPR-PR-OFFSET   PIC 9(5).
            15  QPR-PR-LEN      PIC 9(5).
            15  QPR-PR-FORMAT   PIC X.
                88  QPR-PR-NUMBER   VALUE "N".
                88  QPR-PR-BINARY   VALUE "B".
                88  QPR-PR-DECIMAL  VALUE "D".
            15  QPR-PR-ACTION   PIC X.
                88  QPR-PR-REDRAW   VALUE "R".
                88  QPR-PR-RECREATE VALUE "C".
            15  QPR-PR-VAR-ACT  PIC X.
                88  QPR-PR-RESET-LEN
                                    VALUE "L".
        10  FILLER              PIC X(4).
******* QPR-PR-VAR-DATA ********
        10  QPR-PR-VALUE        PIC X(2000).
```

```
           10  QPR-PR-NUM-VALUE REDEFINES QPR-PR-VALUE
                                PIC 9(5).
           10  QPR-PR-BIN-VALUE REDEFINES QPR-PR-VALUE.
               15  QPR-PR-BIT-VALUE OCCURS 8
                                PIC X.
```

PR-ID

The id of the object to be accessed.

PR-ROW

The row of the static object to be accessed.

PR-COL

The row of the static object to be accessed.

PR-KEY

The key of the property to be accessed.

PR-OBJECT-TYPE

The type of the object to be accessed.

P =	form
S =	static
F =	field
G =	group
R =	repeat

PR-TYPE

The type of the property to be accessed.

L =	length
N =	numeric
C =	character
V =	variable

PR-VAR-TYPE

The type of variable property to be accessed.

A thru J depending on the position of the property in the variable portion of the record.

PR-OFFSET

The zero-based offset of the property within the area defined by PR-TYPE and PR-VAR-TYPE.

PR-LEN

The length of the property.

PR-FORMAT

The format of the property value passed or returned.

Blank =	character in PR-VALUE
N =	numeric in PR-NUM-VALUE
D =	one byte value converted to numeric in PR-NUM-VALUE
B =	one byte value converted to 8 on/off (1/0) values in PR-BIN-VALUE

PR-ACTION

Special action to be taken during function processing.

R =	redraw
C =	recreate
T =	retrieve from client (only referenced in thin client)

PR-VAR-ACT

Special action to be taken if accessing a variable property.

L =	reset length of property value to PR-LEN

PR-VALUE

The value of the property.

CHAPTER E15
Miscellaneous parameters

Overview

These parameters are used with various functions. The NAME-DEF parameter is used by functions needing to pass just the name of a form or form related object. he NULL-PARM parameter is used by functions not needing to pass any data. The BUFFER parameter is used by functions needing to pass a single block of data.

Parameter layouts

```
01  QPR-NAME-DEF.
      05  QPR-ND-RET-CODE          PIC S9(4) COMP-5.
      05  QPR-ND-NAME              PIC X(8).

01  QPR-NULL-PARM.
    05  QPR-NP-RET-CODE           PIC S9(4) COMP-5.

01  QPR-BUFFER.
    05  QPR-BF-RET-CODE           PIC S9(4) COMP-5.
    05  QPR-BF-LEN                PIC S9(4) COMP-5 VALUE +80.
    05  QPR-BF-DATA               PIC X(80).
```

ND-NAME

The name of the form or object to be accessed.

BF-LEN

The length of the data to be passed.

BF-DATA

The data to be passed.

APPENDIX

APPENDIX A
Configuring COBOL FormPrint

The COBOL FormPrint runtime checks for a number of configuration variables in order to control various aspects of its behavior. These variables may be assigned to corresponding environment variables or may be placed in a configuration file. It is usually more convenient to place them in a file because this makes it easier to change the variables. The runtime looks for a configuration file by checking for the following (in the order listed):

1. A file called "QPRMYOWN.CFG".
2. The file referenced by the environment variable QPRCFG.
3. A file called "QPR.CFG".

To set up an environment variable, use the command that is appropriate for the system that you are running on. If you are using a configuration file, set up the variables in the following format:

QPR???=?

Please note that the configuration variable must always be:

1. in upper-case
2. one variable per line

The following is a list of the variables used:

- QPRCFG · the name of the configuration file. If this variable is not defined, the runtime will check for a default configuration file called "QPR.CFG". To check for a default configuration file in the current directory first and then some other file, set this variable to something like ".;\QPR\myQPR.cfg". In order to prevent problems if multiple QPR systems are being used and each system is relying on QPRCFG, a private configuration file called QPRMYOWN.CFG will be searched for in the current directory before anything else.
- QPRDEC · set to 1 for "decimal-point is comma"
- QPRDIR · list of directories containing form, font and icon files
- QPREBC · set to 1 to indicate that EBCDIC signs are being used.
- QPREUR · If you are using QPRIBM=1, set QPREUR=xxx where xxx is the numeric representation (128-255) of the character in the IBM character set that you wish to use for the euro symbol (ansi 128).
- QPRFON · set to xxxxxxxx where x is 0 (off) or 1 (on) as follows:
  - x1 forces 1.x compatibility mode.
  - x2 adjusts screen font width before selecting font to pick widest font possible and help with consistent font selection.
  - x3 inserts extra spaces between printed characters to achieve best font match in terms of width. This should be set if you are using a font width increment.
  - x4 outputs details of selected fonts to qpr.log file
  - x5
  - x6
  - x7 minimizes variations in a font across printers and operating systems. It does this by causing fonts to be

selected so that a certain string of characters fits exactly in an area delimited by the font Width multiplied by the number of characters in the string. The string defaults to the 17 characters "abcdeghknopqsuxyz" but can be reset using the configuration variable QPRSTR. Extra pixels will be added to each character if necessary to match the target width.

- x8 adds extra pixels to selected screen characters so that screen output matches print output more closely. x7 switch must also be set for x8 to take effect.

QPRIBM · set to 1 to indicate that your program is handling data in ASCII (IBM-PC) rather than ANSI format

QPRLOG · force COBOL FormPrint to create a log file. The name of log file will be QPR.LOG. The log file will not be deleted by COBOL FormPrint. To remove it, you must use an operating system command

QPRMTH · for compatibility with earlier versions of COBOL FormPrint

QPROHT · set to 640 to have the Print Preview window scaled up to fill more or less the whole screen (use in conjunction with QPROWD).

QPROIC · set to 1 to have bitmaps scaled appropriately in the Print Preview window (use in conjunction with QPROHT and QPROWD).

QPROWD · set to 480 to have the Print Preview window scaled up to fill more or less the whole screen (use in conjunction with QPROWD).

QPRPVT · set this to the name of a file containing the text to be displayed in Print Preview. The file should be a regular text file laid out as follows (leave the ids intact and change the text as appropriate):

```
001~File
021Get ~all pages
022E~xit preview
023~Cancel print
024~Print…
002~View
041~Next page  ^PgDn
042~Previous page  ^PgUp
043~First page
044~Last page
003~Edit
061~Find…  F3
900Print preview
901Cancel print
902Are you sure?
903Exit preview
904Would you like to print previewed pages?
905Progress...
906Getting page
907Printing page
908String not found
909(ScrollUp)
910(ScrollDn)
911(ScrollLt)
912(ScrollRt)
913myrange.pan
```

QPRPVW
- set to 1 to automatically invoke Print Preview mode.
- set to 5 to automatically invoke Print Preview mode for dynamic forms or multi-form pages.
- set to 9 (1 + 8) or 13 (5 + 8) to display a progress window during retrieval and printing of pages.

QPRSGN
- set to 1 and all signed fields will be assumed to have a separate byte for the sign. If you wish to use this option and have existing panels with signed fields, you must run each panel through the generator in order for FD-PROG-LEN to be recalculated (incremented by 1).

QPRSTR
- used in conjunction with QPRFON x7 switch to specify the string used to check a font's size. Defaults to "abcdeghknopqsuxyz".

QPRTMO
- Set to the number of minutes after which Print Preview should timeout if there has been no user activity. Form-key will be set to -1 in your program.

APPENDIX B
Form File Organization

It is intended that forms files should contain as large a number of forms as you wish. If you wish to have multiple forms files containing logical groups of forms, this is quite permissible. You can have up to three forms files open at any one time. It is NOT recommended that you should have a forms file for each form.

Free space within a forms file that results from updating forms definitions may not always be reused. This may cause your forms file to grow much larger than is necessary. A utility program is included to compress forms files. Invoke this program as follows:

QPRCHECK FORMS-FILE-NAME

This will create a new file called CHECKED.PAN which can be copied back to the original file and then deleted. If QPRCHECK stops and reports an error concerning a forms, please contact your technical support representative.

QPRCHECK can also be used to update a forms file as follows:

```
QPRCHECK SOURCE-FORM-FILE TARGET-FORM-FILE FORM-NAME(S)
```

In this example, the TARGET-FORM-FILE will be created if it doesn't exist.

A fourth parameter may also be passed to QPRCHECK. If the fourth parameter is used the form(s) being moved will also be renamed to the name(s) specified in the fourth parameter.

APPENDIX C
Return Codes

Code	Description
1	not found
8	printer not selected
9	printer not selected
12	form not found
-1	print preview timed out
-10	printer list returned
-17	print preview cancelled
-18	print preview exited without printing
-24	print preview exited with printing

APPENDIX D
The Code Generator

The code generator is invoked by selecting the File/Generate menu option. Code will be generated for the form being edited.

As mentioned in Chapter C2, code is generated based on a template file called UIB.CBX. The editor prompts for the name of the template file to be used. The last used template file is the default. A template file can be modified to produce code that may be more suited to your own needs. Within the file, there are a number of occurrences of the "$" character followed by 3 numeric digits. These are generator tokens and have special meaning when the generator runs. The meaning of each token is as follows:

000 generate a new file
001 form name
002 length of program Fields area
003 length of program Colors area
004 used internally
005 used internally
006 used internally
007 for all fields with Program length property not zero, do the following...
008 end for (like end-perform)
009 field name derived from Name property
010 field format derived from Format property
011 if field is member of a repeat group
012 else (if not member of repeat group)
013 repeat group Vertical occurrences
014 for all fields in this repeat group, do the following...
015 end repeat
016 end of 011 processing
017 value of field Id property
018 for all fields with Program number property not zero, do the following...
019 for all fields except repeat fields with Occurrence property > 0, do the following...
020 display number of lines generated
021 only generate the first time thru this for loop
022 fi-name (form file name)
023 if this field, color or type does not immediately follow previous item, do the following... (REDUNDANT)
024 difference in bytes between start of this item and end of last item
025 end of 023 processing (REDUNDANT)
026 value of field property, expressed as 026x(y) where x is the property id (see below) and y is the length of the data
027 simple conditional, expressed as:
027x=(y) = if x equal y
027x!=(y) = if x not equal y
027x>(y) = if x greater than y
027x<(y) = if x less than y
027x>=(y) = if x greater than or equal y
027x<=(y) = if x less than or equal y
x and/or y may be constants or dollar variables
028 end of condition
029 generate a new file but issue warning if it already exists
030 value of group property, expressed as 030x(y) where x is the property id (see below) and y is the length of the data
031 else (use with 027 and 028)
032 value of form property, expressed as 032x(y) where x is the property id (see below) and y is the length of the data
033 for all statics, do the following...
034 value of static field property, expressed as 034x(y) where x is the property id (see below) and

y is the length of the data
035 express dimension in decicells as whole cells with 1 decimal - must be immediately followed by a 026, 030, 032, 034 or 054
036 dollar sign
037 position in cells incremented by 1 (same format as 035)
038 width in 1/10 cells in terms of pixels (same format as 035)
039 height in 1/10 cells in terms of pixels (same format as 035)
040 (thru 049) user variables (may be numeric or non-numeric)
050 math functions - x must be a user variable
050x=(y) = move y to x (y may be numeric or non-numeric)
050x+(y) = compute x = x + y (x and y must be numeric)
050x-(y) = compute x = x - y (x and y must be numeric)
050x*(y) = compute x = x * y (x and y must be numeric)
050x/(y) = compute x = x / y (x and y must be numeric)
050x%(y) = divide x by y remainder x (x and y must be numeric)
050x#(y) = compute x = y converted to numeric (eg #(A) is 65)
050x&(y) = compute x = x ended with y
051 loop continuously
052 break out of loop
053 Endloop
054 used internally
055 used internally
056 used internally
057 for all groups, do the following...
058 foreground color of form, field or static depending on context
059 background color of form, field or static depending on context
060 used internally
061 used internally
070 (thru 079) more user variables (may be numeric or non-numeric)
080 special color code (same format as 035)
081 special color code for fg (same format as 035)
082 special color code for bg (same format as 035)
083 row in cells expressed in logical cells with 2 decimals
084 column in cells expressed in logical cells with 2 decimals
085 width in 1/10 cells expressed in logical cells with 2 decimals
086 height in 1/10 cells expressed in logical cells with 2 decimals
087 decimal point symbol
088 width in cells expressed in logical cells with 2 decimals
089 height in cells expressed in logical cells with 2 decimals
090 initial value in display format
091 for all fields and statics in sequence, do the following…
092 used internally
092 used internally
093 used internally
094 used internally
095 used internally
096 used internally
097 text with all ~ and leading spaces removed (same format as 035)
098 group name derived from Name property
099 used internally
100 generate space and low-values in hexadecimal
101 Font
102 right parenthesis
103 used internally
104 suppress end-of-line
105 used internally

106 for all repeat groups, do the following...
107 value of item in repeat group record (same format as 026)

A number of the tokens also require a property id to be specified. These tokens are as follows:

26 - value of field property (uses field property id)
30 - value of group property (uses group property id)
32 - value of form property (uses form property id)
34 - value of static property (uses static property id)

The property ids are all listed in Appendix E.

APPENDIX E
Property Ids and Keys

Form Properties

Id(Len)	Key	Name
L8(2)	PL-0000800002N	Length of Description
L10(2)	PL-0001000002N	Length of Title
N8(2)	PN-0000800002N	Field count
N10(2)	PN-0001000002N	Program field count
N12(2)	PN-0001200002N	Program field length
N28(2)	PN-0002800002N	Total width
N30(2)	PN-0003000002N	Total height
N34(2)	PN-0003400002N	Cell width
N36(2)	PN-0003600002N	Cell height
N38(2)	PN-0003800002N	Font id
C0(8)	PC-0000000008	Name
C22(1)	PC-0002200001	Blank numerics
C24(1)	PC-0002400001	Format numerics
C26(1)	PC-0002600001	Initialize numerics
C29(1)	PC-0002900001D	Cell height increment
C30(1)	PC-0003000001B	Miscellaneous
C38(1)	PC-0003800001D	Color
C39(1)	PC-0003900001B	Program date format
C66(1)	PC-0006600001B	More options
C67(1)	PC-0006700001B	Options-3
VA0(*)	PVA00000*****--L	Description
VB0(*)	PVB00000*****--L	Title

* the length of the property value depends on the corresponding length property

Field Properties

Id(Len)	Key	Name
L8(2)	FL-0000800002N	Length of Format
L10(2)	FL-0001000002N	Length of Caption
L12(2)	FL-0001200002N	Length of Value
L18(2)	FL-0001800002N	Length of Discrete values
N0(2)	FN-0000000002N	Id
N6(2)	FN-0000600002N	Occurrence
N8(2)	FN-0000800002N	Base id
N10(2)	FN-0001000002NC	Row
N12(2)	FN-0001200002NC	Column
N14(2)	FN-0001400002N	Program offset
N20(2)	FN-0002000002N	Program number
N22(2)	FN-0002200002NC	Width
N24(2)	FN-0002400002NC	Height
N26(2)	FN-0002600002N	Maximum length
N28(2)	FN-0002800002N	Program length
N30(2)	FN-0003000002N	Item length
N38(2)	FN-0003800002N	Group id
N40(2)	FN-0004000002N	Repeat id
N42(2)	FN-0004200002NR	Font id
C0(30)	FC-0000000030	Name

C30(1)	FC-0003000001-C	Type
C31(1)	FC-0003100001	Protection
C32(1)	FC-0003200001D	Decimals
C36(1)	FC-0003600001	Initialize if numeric
C37(1)	FC-0003700001B	Miscellaneous
C58(1)	FC-0005800001	Justify
C59(1)	FC-0005900001	Fill character
C61(1)	FC-0006100001	Special format
C65(1)	FC-0006500001	Usage option
C68(1)	FC-0006800001	Blank if zero
C69(1)	FC-0006900001-C	Control type
C70(1)	FC-0007000001DR	Color
C72(1)	FC-0007200001-R	Border type
C73(1)	FC-0007300001B	Program data
VA0(*)	FVA00000*****--L	Format
VB0(*)	FVB00000*****-RL	Caption
VC0(*)	FVC00000*****-RL	Value
VE0(*)	FVE00000*****--L	Discrete values

* - the length of the property value depends on the corresponding length property

Static Properties

Id(Len)	Key	Name
L8(2)	SL-0000800002N	Length of Text
N0(2)	SN-0000000002N	Id
N2(2)	SN-0000200002NC	Row
N4(2)	SN-0000400002NC	Column
N6(2)	SN-0000600002NC	Width
N8(2)	SN-0000800002NC	Height
N10(2)	SN-0001000002NR	Font id
C0(1)	SC-0000000001DR	Color
C1(1)	SC-0000100001-C	Type
C2(1)	SC-0000200001	Justify
C3(1)	SC-0000300001B	Miscellaneous
VA0(*)	SVA00000*****-RL	Text

* - the length of the property value depends on the corresponding length property

Group Properties

Id(Len)	Key	Name
L6(2)	RL-0000000002N	Length of variable data
N0(2)	GN-0000000002N	Id
N2(2)	GN-0000200002NC	Row
N4(2)	GN-0000400002NC	Column
N6(2)	GN-0000600002NC	Width
N8(2)	GN-0000800002NC	Height
N10(2)	GN-0001000002N	Field count
C0(30)	GC-0000000030	Name
C33(1)	GC-0003300001	Select type
C44(1)	GC-0004400001DR	Color
C46(1)	GC-0004600001-R	Border
VA0(*)	GVA-00000*****-CL	Field ids

* - the length of the property value depends on the corresponding length property

Repeat Group Properties

Id(Len)	Key	Name
L8(2)	RL-0000000002N	Length of Field ids
N0(2)	RN-0000000002N	Id
N2(2)	RN-0000200002NC	Row
N4(2)	RN-0000400002NC	Column
N6(2)	RN-0000600002NC	Width
N8(2)	RN-0000800002NC	Height
N10(2)	RN-0001000002N	Vertical occurs
N14(2)	RN-0001400002NC	Vertical gap
N22(2)	RN-0002200002N	Horizontal occurs
N26(2)	RN-0002600002NC	Horizontal gap
N40(2)	RN-0004000001N	Program field length
N42(2)	RN-0004200001N	Field count
C0(1)	RC-0000000001-R	Border type
C2(1)	RC-0000200001B	Miscellaneous
VA0(*)	RVA-00000*****-CL	Field ids

* - the length of the property value depends on the corresponding length property

INDEX

INDEX